Designing Next Generation Test Systems

An In-Depth Developers Guide

NATIONAL INSTRUMENTS™

Designing Next Generation Test Systems

An In-depth Developers Guide

NATIONAL INSTRUMENTS™

Contents

Chapter 5
Instrument Bus Performance – Making Sense of Competing Bus Technologies for Instrument Control

Chapter 6
Understanding a Modular Instrumentation System for Automated Test

Chapter 7
PXI – The Industry Standard Platform for Instrumentation

Section 3
Strategies for Improving Test System Performance

Chapter 8
Maximizing Throughput in an Automated Test System

Chapter 9
Maximizing Accuracy in Automated Test Systems

Chapter 10
Designing and Maintaining a Test System for Longevity

Section 4
Case Studies and Customer Applications

Chapter 11
Software-Defined Radio Architecture for Communications Test

Chapter 12
Microsoft Uses NI LabVIEW and PXI Modular Instruments to Develop Production Test System for Xbox 360 Controllers

Chapter 13
U.S. Air Force Increases Mission-Capable Rates with PXI

Chapter 14
Sanmina-SCI Exceeds Throughput Goals with PXI Tester and Multithreaded Software

Executive Summary

1

Designing Next Generation Test Systems

Welcome to the Designing Next Generation Test Systems Developers Guide. This guide is the first in a series of whitepapers designed to help you develop test systems that lower your cost, increase your test throughput, and can scale with future requirements.

Test managers and engineers use automated test systems in applications ranging from design validation to end-of-line production test to equipment repair diagnostics with the goal of ensuring the quality and reliability of a product that reaches the end customer. They can use automated test systems to perform simple "pass" or "fail" tests or a whole range of product characterization measurements. As it grows exponentially more expensive to detect flaws later in the design cycle, automated test systems have quickly become an even more important part of the product development process. This first document titled "Designing Next Generation Test Systems" describes the challenges that continue to pressure engineering teams to reduce the cost and time of test. It also provides insight into how test managers and engineers are overcoming these challenges by building modular, software-defined test systems that significantly increase test system throughput and flexibility while reducing overall cost.

Today's test engineers face a new set of pressures. Test engineers are presented with the following realities of today's product development environment:

- Product designs are more complex than their previous generations
- Development cycles are shorter to stay competitive and meet customer demand
- Budgets are decreasing while product testing is becoming more expensive

Increasing Design Complexity

One of the most visible trends is the increased device complexity. For example, the consumer electronics, communications, and semiconductor industries continue to drive the convergence of digital imaging/video, high-fidelity audio, wireless communication, and Internet connectivity into a single product. Even the automobile has integrated sophisticated car entertainment and information systems, safety and early warning systems, and body and engine control electronics. Test system designs must be flexible enough to support the wide variety of tests that differ between product models but also scalable to accommodate a larger number of test points as new measurement functionality is required.

Shorter Product Development Cycle

The ever-increasing demand for the latest product or technology combined with the competitive nature of being the first to market puts pressure on design and test engineering teams to shorten their product development cycles. To be successful, engineering teams need to develop new test strategies to decrease test time and improve efficiencies from design through production.

Increasing Test Cost and Decreasing Test Budget

Increase device functionality often leads to a more expensive and time-consuming test process. However, the cost to build each function is decreasing, which challenges test engineering departments to reduce its cost and budget, as shown in Figure 1-1. As a result, engineers must develop test strategies that reduce cost by increasing the throughput of their test systems, reducing the maintenance and upgrade costs, and lowering the required capital investment.

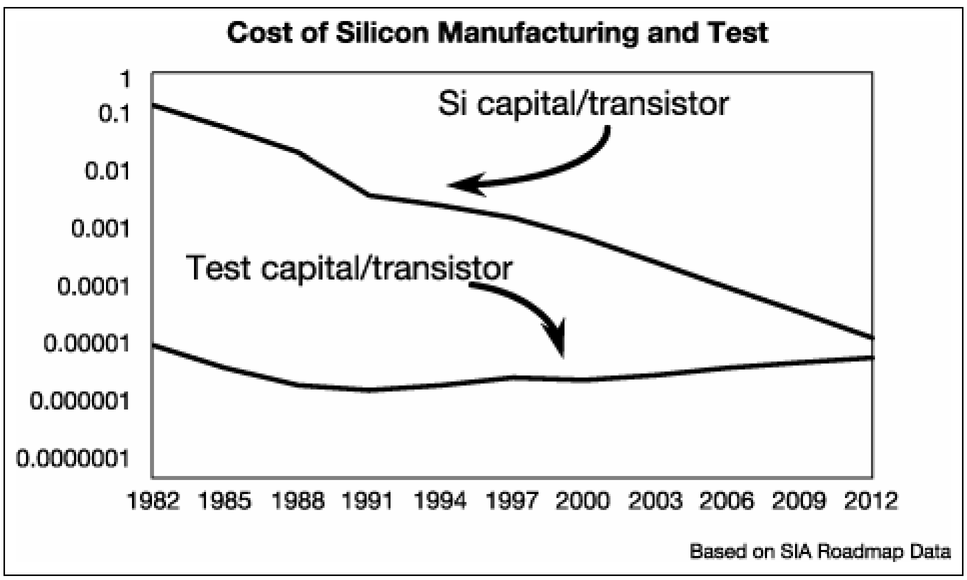

Figure 1-1. Data from SIA illustrates that the cost of silicon (or device function) has decreased over time, but the cost to test continues to increase.

Designing Next Generation Test Systems

To meet the challenges they face with increased device complexity, shorter development cycles, and decreased budgets, test managers and engineers are being forced to abandon traditional test design strategies based on traditional box instruments or "big iron" propriety ATE systems. These stand-alone instruments lack the flexibility needed as the software processing and user interface are defined by the supplier and can only be updated by the supplier through firmware. This makes it difficult to perform measurements not defined in the instrument's firmware and measurements for new standards or to modify the system if requirements change. These devices also lack necessary integration capabilities such as data streaming and synchronization because they are designed primarily to

work as stand-alone instruments and not for integrated system use. Propriety ATE systems, such as highly integrated production chip testers, provide the performance needed but are typically cost-prohibitive and can leave engineering teams vulnerable to obsolescence and untimely system redesigns.

In response to these trends, test managers and engineers are now implementing modular, software-defined test architectures based on widely adopted industry standards to provide:

- Increased test system flexibility deployable to a variety of applications, business segments, and product generations
- Higher-performance architectures that significantly increase test system throughput and enable tight correlation and integration of instruments from multiple suppliers including precision DC, high-speed analog and digital, and RF signal generation and analysis
- Lower test system investments by reducing initial capital investment and maintenance cost while increasing equipment use across multiple test requirements
- Increased test system longevity based on widely adopted industry standards that enable technology upgrades to improve performance and meet future test requirements

National Instruments, a leader in automated test, is committed to providing the hardware and software products engineers need to create these next generation test systems.

This in-depth developers guide includes the information you need to design your next test system architecture. This introduction describes a test system architecture, figure 1-2, that provides engineers with a strategy to meet challenges related to increased device complexity, shorter development cycles, and decreased budgets.

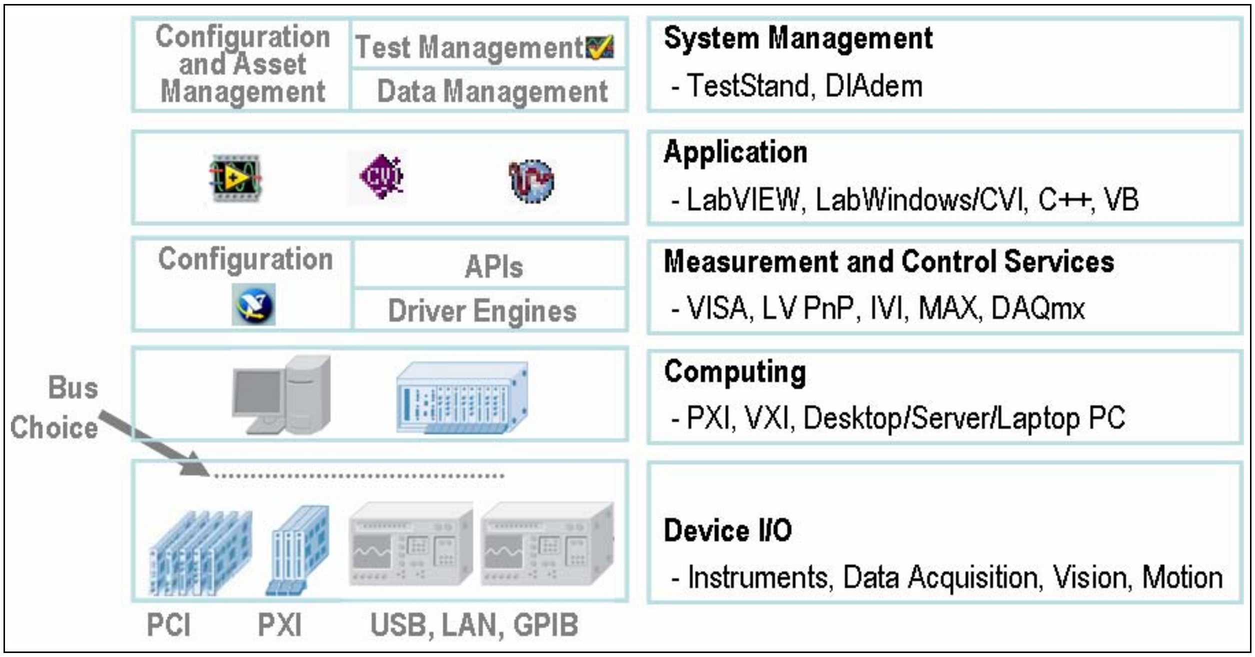

Figure 1-2. National Instruments offers a complete hardware and software solution for designing automated test systems.

Test System Management Software

An automated test system requires the implementation of several tasks and measurement functions – some specific to the device under test (DUT) and others repeated for every device tested. To minimize maintenance costs and ensure test system longevity, it is important to implement a test strategy that separates the DUT-level tasks from the system-level tasks so engineers can quickly reuse, maintain, and change test programs (or modules) created throughout the development cycle to meet specific test requirements.

In any test system, there are often operations that are different for each device tested and operations that are common for each device tested such as system-level tasks.

Operations different for each device	Operations common for each device
Instrument configuration	Operator interfaces
Measurements	User management
Data acquisition	DUT tracking
Results analysis	Test flow control
Calibration	Storing results
Test modules	Test reports

Some companies have written their own test executives and spent valuable engineering resources to develop test management software from the ground up. This often results in reduced productivity and ties up resources over time for software maintenance. To maximize productivity, engineering teams should use commercially available test management software, such as NI TestStand, to reduce the development of operations that are common for each device. By using this software, engineers can focus their development efforts on the operations that are different for each device.

For more information, view the whitepaper titled, "Developing a Modular Software Architecture."

Application Development Software

The application development environment (ADE), such as National Instruments LabVIEW and LabWindows/CVI, plays a critical role in test system architectures. With these tools, test system developers can communicate to a variety of instrumentation, integrate measurements, display information, connect with other applications, and much more. Ideally, the ADE(s) used to develop test and measurement applications provide ease of use, compiled performance, integration of a diverse set of I/O, and programming flexibility to meet the requirements for a range of applications.

Ease of use goes beyond how quickly someone can get up and running. With easy-to-use ADEs, developers can easily integrate processing routines with multiple measurement devices, create sophisticated user interfaces, deploy and maintain applications, and modify the application as product designs evolve and system needs expand.

For more information, view the whitepaper titled, "Choosing the Right Software Application Development Environment."

Measurement and Control Services

Measurement and control services play an important role by providing connectivity to various hardware assets in the system, system configuration, and diagnostic tools. For example, NI Measurement & Automation Explorer (MAX) automatically detects hardware assets including data acquisition and signal conditioning hardware; GPIB, USB, and LAN-controlled instruments; PXI systems; VXI devices; and modular instrumentation so developers can configure them all in one place. Integrated diagnostic tests ensure that devices function properly, and test panels provide a quick way to check the functionality of the hardware before developers begin programming. Measurement and control services should also provide integration with the application development software layer through application programming interfaces (APIs) so developers can easily program their instruments. In fact, the components of this services software – hardware drivers, application programming interfaces (APIs), and a configuration manager – must seamlessly integrate within the ADEs to maximum performance, increase development productivity, and reduce overall maintenance.

For more information, view the whitepaper titled, "Developing a Modular Software Architecture."

Computing and Measurement Bus

At the center of every automated test system today is a computer in the form of a desktop PC, server workstation, laptop, or embedded computer as used with PXI and VXI. An important aspect of the computing platform used is the ability to connect (and communicate) to the wide variety of instruments in a test system. There are several different instrumentation buses available for stand-alone and modular instruments including GPIB, USB, LAN, PCI, and PCI Express. These buses have differing strengths making some more suitable for certain applications than others. For example, GPIB has the widest adoption for instrument control and wide availability of instrumentation; USB provides wide availability, easy connectivity, and high throughput; LAN is well-suited for distributed systems; and PCI Express provides the highest performance.

The widespread use of the PC has generated the proliferation of high-performance internal buses including PCI and PCI Express, which provide the lowest latency and highest data throughput or bandwidth. The PCI bus provides up to 132 MB/s of bus bandwidth and PCI Express, an evolution of PCI, can scale up to 4 GB/s to meet growing bandwidth needs while still providing complete software compatibility with PCI. Figure 1-3 illustrates the latency and bandwidth performance of the most popular instrument control buses.

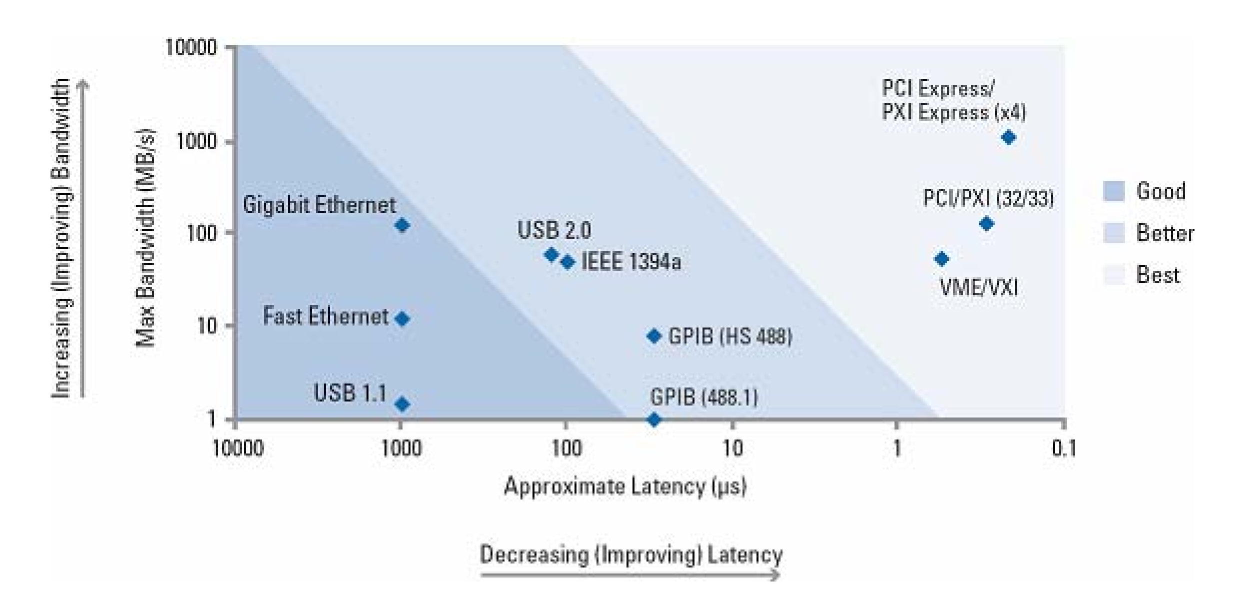

Figure 1-3. A comparison of various instrument control buses. PCI and PCI Express provide the best bandwidth and latency or overall throughput performance.

For more information, view the whitepapers titled: "Hybrid Systems: Integrating Your Multi-Vendor, Multi-Platform Test Equipment" and "Instrument Bus Performance: Making Sense of Competing Bus Technologies for Instrument Control."

Measurement and Device I/O

Fundamentally, there are two types of instrumentation architectures today – traditional and virtual. Figure 1-4 illustrates the similarities in these two approaches. Both have measurement hardware, a chassis, a power supply, a bus, a processor, an OS, and a user interface.

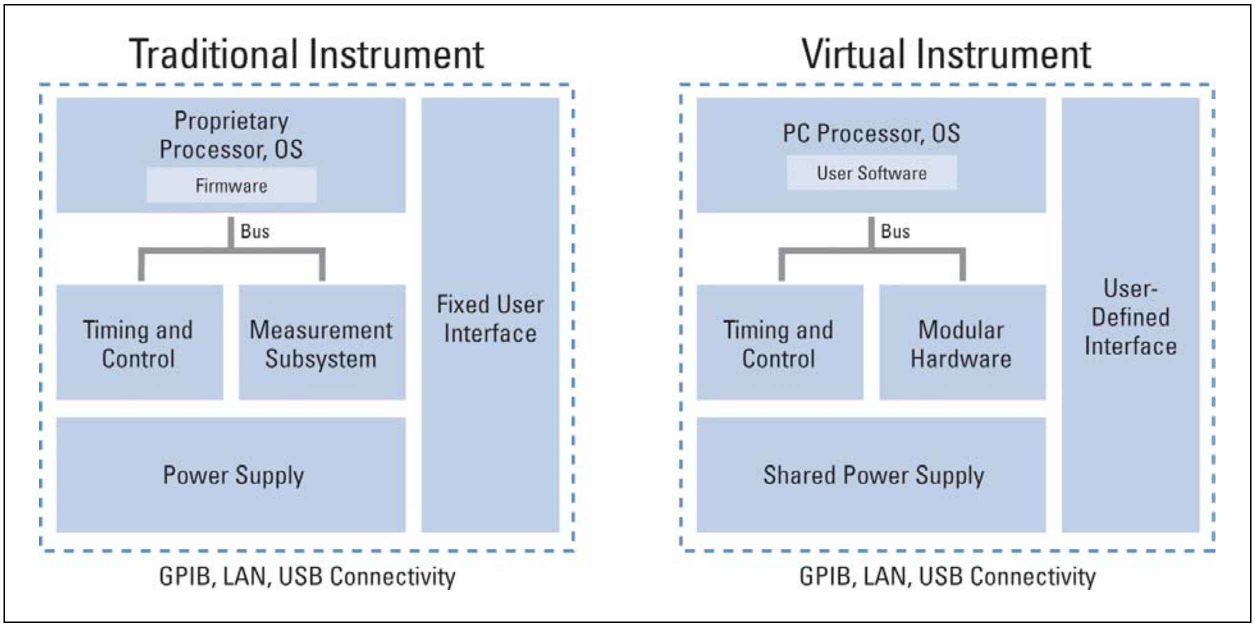

Figure 1-4. Traditional and virtual instrumentation architectures share similar hardware components; the primary difference between the architectures is where the software resides and whether it is user-accessible.

The most obvious difference from a hardware standpoint is how the components are packaged. A traditional, or stand-alone, instrument puts all of the components in the same box for every discrete instrument. The measurement functionality, analysis, displays, and control of the instruments is defined by the supplier.

By contrast, modular, software-defined virtual instruments incorporate general-purpose measurement hardware that helps users go beyond the standard capabilities and define their own measurements and user interfaces in software. With a modular approach, engineers can define the test system measurement functionality and build systems that scale to meet future demands. Through a modular, software-defined approach, users make custom measurements, perform measurements for emerging standards, or modify the system if requirements change (for example, to add instruments, channels, or new measurements). This combination of flexible, user-defined software and scalable hardware components is the core of modular instrumentation.

For more information, view the whitepapers titled: "Understanding a Modular Instrumentation System for Automated Test" and "PXI: The Industry Standard Platform for Instrumentation."

Conclusion: Designing Next Generation Test Systems

Increased device complexity, shorter development cycles, and decreased budgets provide an opportunity for engineering teams to re-evaluate their current automated test strategies and look for areas to increase efficiency and reduce cost. When designing next generation test systems it is important to incorporate strategies that increase system flexibility, deliver higher measurement and throughput performance, lower test system cost, and expand longevity. Modular, software-defined automated test systems overcome the shortfalls of past solutions based on stand-alone instrumentation or cost-prohibitive propriety ATE systems. A modular hardware platform based on widely adopted industry standards platform, such as PXI, allow engineers to develop scalable test systems that tightly integrate the functionality from a variety of instrumentation suppliers. In addition, it also allows engineering teams to integrate current equipment investments lowering the initial cost of implementation. Along with software-defined measurements that make use the latest PC technology such as multiple core processors and PCI Express, next generation test systems can significantly improve throughput performance and scale to meet the demands of different product generations and business segments.

Several companies already implement a modular, software-defined test system strategy and prove the return on their investment. For example, Microsoft developed the test system for the Xbox 360 controllers based on NI LabVIEW and PXI Modular Instruments that resulted in a test system that operates twice as fast that their previous generation. The U.S. Air Force developing test architecture that supporting their premier fighter aircrafts. Using a PC-based software and hardware architecture, they were able to lower costs and reduce size of test systems by 50 percent. Sanmina-SCI builds FDA approved medical device test systems based on NI TestStand and PXI that exceeds their requirements to test over 83,000 devices per week and exceeds their yield production requirements by 95 percent.

Guidelines for Designing Next Generation Test Systems

2

Developing a Modular Software Architecture for Reducing Development Cycles and Cost

Industry Trends and Challenges

The rapid advancement in global design and new product development are creating new opportunities for test managers and engineers to develop test systems that greatly contribute to accelerating the product development cycle. The approach to test system development is shifting from using application-specific, turnkey test systems to focusing on building a modular test framework. The next-generation test framework should accommodate the test requirements of multiple products and facilitate the addition of new test technologies for additional test coverage in the future. Developing a robust, open test software framework is a key component of designing the architecture of a modular test framework.

The need to develop a modular, flexible test software framework has become more evident as the ratio of development costs versus capital costs increases. Development costs, such as system integration and software development, are often 2 to 10X more than capital costs in most test systems today. This rapid rise in development costs is causing a profound impact on the ability for many test engineering groups to keep up with the accelerated product development cycle and globalization of design and test. The current approaches to test system development are generating far too many projects for today's engineers to keep up with.

The complexity of products that engineers need to test is increasing rapidly as well. Markets are demanding higher quality and more features. Currently, a product might include significantly more functionality condensed in the same amount of space. Testing this increasing functionality requires the addition of new technologies to your test system, which can be a challenge for turnkey test systems due to their lack of flexibility. In contrast, a modular test architecture can inherently add new features as they are required. Using a modular test software architecture helps you reduce development costs, decrease product development cycles, and keep up with the increasing complexity of new products.

Defining a Modular Test Software Framework

Designing efficient test systems requires a modular software architecture and development tools optimized for test. To develop test systems faster and more cost-effectively, it is critical that you evaluate your test software architecture to maximize code reuse. Examining your test software architecture consists of evaluating the software development tools you are using and studying your

approach to test code development. Understanding the importance of modular test software architectures and how to develop your tests as modules rather than building stand-alone applications will significantly improve your test software reuse.

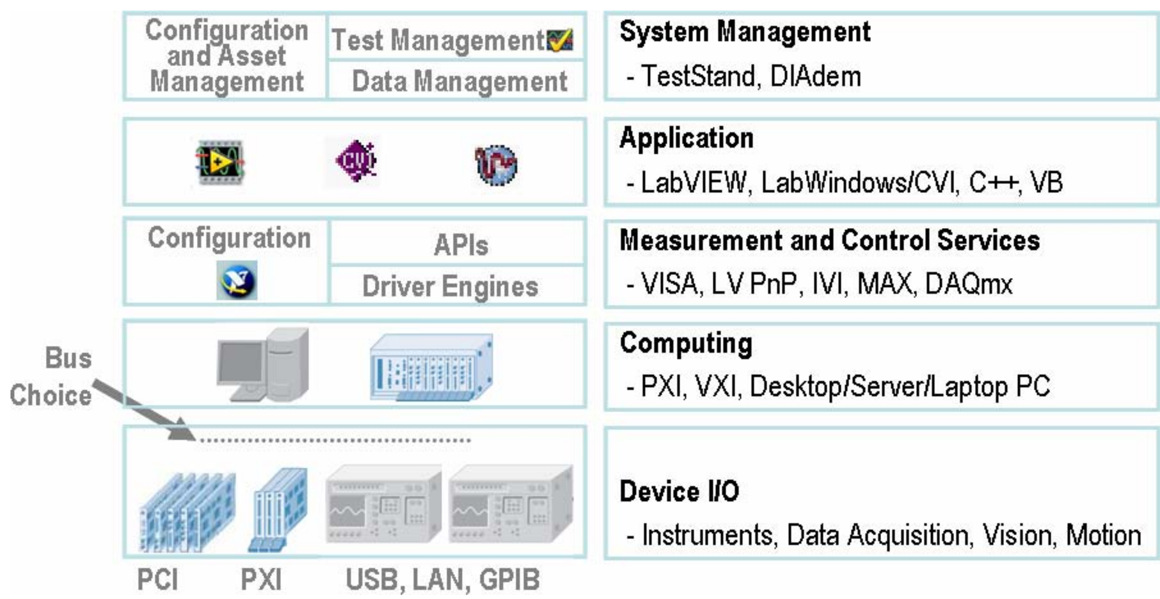

Figure 2-1. Test System Architecture

Incorporating modular test software architectures begins with choosing a software development environment that is designed for easily connecting to your instruments and quickly performing any type of measurement and analysis required for your tests. Such test software development tools include NI LabVIEW, LabWindows/CVI, and Measurement Studio for Visual Studio .NET. Using the proper test development environment, you can more easily share your test programs with others in your group and across test departments in your organization.

Selecting test and measurement hardware with robust software interfaces is another important layer in defining modular software architectures. Measurement and control services software such as NI Measurement & Automation Explorer (MAX), NI-DAQ, Virtual Instrument Software Architecture (VISA), LabVIEW Plug&Play drivers, and Interchangeable Virtual Instrument (IVI) drivers provide modular hardware interfaces for configuring and programming your test system through the use of virtual channel names, virtual devices, and simulation interfaces. Such modular measurement and control services driver software helps you avoid developing test programs that are permanently tied to specific hardware and channels in your test system, thus increasing the ease of code reuse.

Test modules created by design engineers for prototype and validation tests are an integral asset to production test departments. Production test engineers can easily integrate tests developed during the prototype and validation phases into final

automated test systems using industry-standard test management software such as NI TestStand.

NI TestStand provides built-in test management capabilities such as test module adapters for calling tests written in common test languages such as LabVIEW, LabWindows/CVI, C/C++, and Visual Studio .NET regardless of the function prototypes defined for each test. Maximizing code reuse between the product design and manufacturing teams reduces test development effort ensuring production schedules are met and demands for improved quality are upheld. The flexible NI TestStand module adapters ensure maximum code reuse across the product development cycle with minimal training and code

The National Instruments Modular Test Software Framework

The management level of the modular test software framework is responsible for directing the execution of the whole test system. The open software architecture of NI TestStand is a popular choice for test management software and greatly reduces the effort required to implement a scalable test software framework. NI TestStand provides a completely modular and open architecture that can be used "as is" off the shelf or used as individual components for designing your completely customized NI TestStand-based test systems. Figure 2-2 describes the NI TestStand architecture graphically.

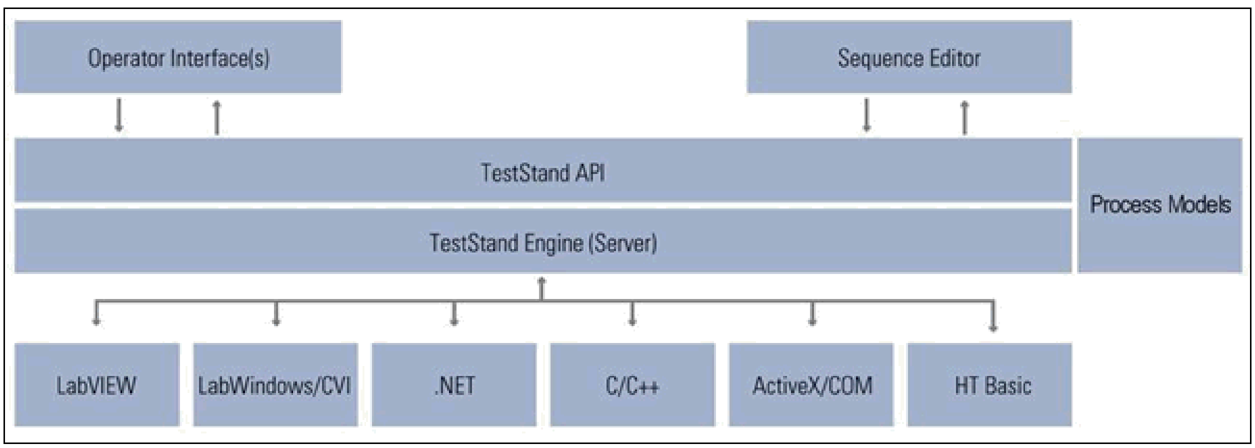

Figure 2-2. NI TestStand Test Management Software Architecture

The center of the NI TestStand architecture is the NI TestStand Engine, a powerful multithreaded test engine with complete and extensively documented API. By communicating with the NI TestStand Engine, the module adapters provide an open language interface to automate tests written in any language. The process models provide superior modularity between the test code and the system level functions that must be performed. The Sequence Editor provides an easy-to-use, powerful development environment for test sequences. Lastly, the operator interfaces are provided in source code in multiple programming languages for rapid customization to match your exact needs.

NI TestStand Engine

All parts of the NI TestStand architecture are directed by the NI TestStand Engine, which is a set of libraries that export an ActiveX/COM API. The API gives developers the ability to perform any operation on the NI TestStand Engine programmatically by using the more than 1,400 exported functions. The NI TestStand Engine implements multithreading, which enables you to increase throughput by testing multiple units simultaneously. The engine also relieves developers from including limit testing in their test code by implementing this functionality natively. By not including limit testing, test code becomes more flexible and reusable. Another feature of the NI TestStand Engine is that it implements flow control functionality much like any programming language. Finally, it also increases the security of your test system by implementing multilevel user access and management.

Module Adapters

For the NI TestStand Engine to call code written in different languages, it takes advantage of the different module adapters available with NI TestStand. The module adapters provide an open language interface between the NI TestStand Engine and your test code written in LabVIEW, LabWindows/CVI, .Net, C/C++ DLLs, ActiveX/COM and HT Basic. By calling code in different languages, you are able to reuse any existing legacy code as well as take advantage of newer technologies. With code modules, you can send and receive information from your code modules using an arbitrary number of parameters or by using the NI TestStand API. Other features provided by the module adapters include stepping into code modules for debugging and using code templates to improve programmer productivity.

Process Model

Testing a UUT requires more than just executing a set of tests. Usually, the test system must perform a series of operations such as identifying the UUT, logging results, and generating a test report. The set of such operations and their flow of execution are called a process model. The process models provide superior modularity between the test code and the system level functions that must be performed by implementing these system level functions and using them with multiple test sequences. NI TestStand is shipped ships three process models, which can be used as is or fully customized. The Sequential process model can be used for testing one unit at a time, while the Batch and Parallel process models use the NI TestStand multithreading functionality to test more than one unit at the same time.

Sequence Editor

The Sequence Editor provides test engineers all the functionality and tools needed to develop the most sophisticated automated test systems. In the Sequence Editor, Test Sequence Files can be created, debugged and modified. Test Sequence Files contain test steps that can contain code modules developed in any test programming language. Furthermore, the Sequence Editor includes a utility to build deployment packages to ease the distribution of Test Sequences and operator

interfaces. The Sequence Editor also provides user-management services that can prevent some users from using restricted functionality based on the privileges defined by your TestStand administrator. Figure 2-3 shows the Sequence Editor displaying a test sequence written in LabWindows/CVI.

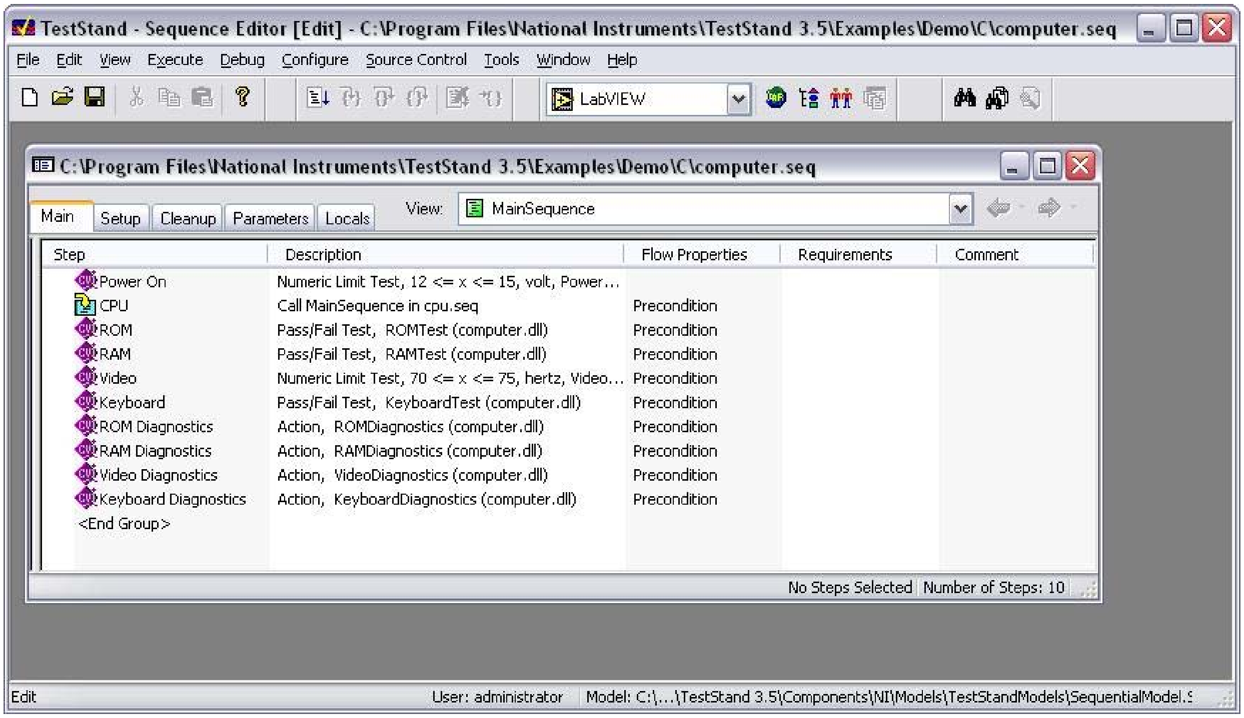

Figure 2-3. NI TestStand Sequence Editor

Operator Interfaces

Lastly, an operator interface is a customizable user interface for NI TestStand that can be used to execute and debug test sequence files created in the Sequence Editor. The operator interface is typically used on a manufacturing floor or if you need to deliver a custom look and feel to your test or validation system. TestStand operator interfaces use the NI TestStand User Interface Controls to facilitate development by fully implementing common features such as Sequence File Display and Execution tracing. NI TestStand includes ready-to-run operator interfaces written in LabVIEW, LabWindows/CVI, C#, VB, and VB .NET.

Application Development Environments (ADE)

ADEs play a critical, visible role in a test software framework. With these tools, the system developer designs and integrates the system that takes measurements, displays information to the end user, connects with other applications, and much more. The ADEs used to develop measurement and automation applications provide an easy-to-use design model, compiled performance, and application-level programming flexibility for a complete range of applications. Equally as important,

these ADEs tightly integrate with Measurement and Control Services Software that connects to a wide variety of I/O devices, and scales from small applications to large systems.

Although it is an important element, ease of use goes beyond how quickly someone can get up and running. With easy-to-use ADEs, developers can easily integrate processing routines with multiple measurement devices, create sophisticated user interfaces, deploy and maintain an application, and modify the application as product designs evolve and system needs expand.

Despite the perceived flexibility benefit of using in-house software architectures, organizations that use proprietary software often incur unintended expenses as they fight to keep up with rapidly advancing technologies, such as OSs and networking technologies, that are not core to their business. This effort can siphon valuable resources away from business operations and often lead to a loss of valuable time. For example, by using an off-the-shelf ADE designed for measurement and automation, developers can quickly and easily upgrade to the latest OS, or integrate emerging Internet and XML standards with minimal development investment. Or, when using an ADE such as Visual Basic or Visual C++, specialized measurement and automation add-ins can significantly reduce development time.

In addition to the tight integration with Measurement and Control Services Software, ADEs used to develop a measurement and automation system can manage and process measurements. To do this most effectively, the ADE incorporates measurement data types directly in the environment so that these measurements are easy to use in additional processing routines. For maximum development productivity, the ADE include comprehensive statistical and numerical analysis functions, as well as high-performance signal processing and control algorithms common in measurement applications. ADEs integrate typical routines found in measurement and automation applications, including functions such as PID and fuzzy logic control, noise reduction, spectral measurements, digital filtering, response measurements, signal detection, numerical integration and differentiation, curve fitting, fractional-octave analysis, and order analysis.

For more information, view the whitepaper titled, "Choosing the Right Software Application Development Environment."

Measurement and Control Services

Selecting test and measurement hardware with robust software interfaces is another important layer in defining modular test architectures. Measurement and control services software such as NI Measurement & Automation Explorer (MAX), NI-DAQ, Virtual Instrument Software Architecture (VISA), LabVIEW Plug&Play drivers, and Interchangeable Virtual Instrument (IVI) drivers provide modular hardware interfaces for configuring and programming your tests. Such modular measurement and control services driver software helps you avoid developing test programs that are permanently tied to specific hardware and channels in your test system, thus increasing the ease of code reuse.

Configuration Manager

A configuration manager, such as MAX, presents a unified system view of measurement hardware supported in the Measurement and Control Services Software. With MAX, users can define channel names to organize signals or specify scaling functions to convert digitized signals to measurement quantities. The key benefit of the configuration manager is the integration with the ADEs. This integration gives developers the ability to integrate multiple measurements easily into a single application without tedious programming. Without these configuration tools, developers must spend needless time configuring these measurement functions programmatically.

Instrument Connectivity

Integrating existing traditional instruments into the test software framework should exploit technologies, such as Plug and Play Instrument Drivers and IVI, to facilitate the communication with these instruments and their interchangeability. A Plug and Play instrument driver is a set of functions, or VIs in the case of LabVIEW, that control a programmable instrument. Instrument drivers help users get started using their instrument from their computer and save them development time and cost because users do not need to learn the programming protocol for each instrument. With open-source, well documented instrument drivers, end users can customize their operation for better performance.

IVI implements a driver framework that facilitates instrument interchangeability. An IVI driver uses a general API for each kind of instrument and separately implements the driver to communicate with particular instruments. By separating the API from the particular driver implementation of each instrument, an engineer can design a system using a particular IVI-compliant oscilloscope; once the system is deployed, he/she can change the brand and model of the instrument without having to rewrite the test application.

Programming Tools

Drivers can go one step beyond providing an easy-to-use API by adding tools to facilitate development by saving you time. I/O assistants are interactive tools for very rapidly creating a measurement or stimulus applications. An example of an I/O assistant is DAQ Assistant, which is part of the NI-DAQmx driver. DAQ Assistant presents a panel to the user for configuring common data acquisition parameters without programming. The combination of easy-to-use assistants and powerful programming environments is necessary to provide both rapid development and the capabilities to meet a breadth of application requirements.

Conclusion: Developing a Modular Software Architecture for Reducing Development Cycles and Cost

A modular software framework consists of three tightly integrated layers, which provide system management, application development and measurement and control services. System management software consists of tools to develop the framework for your whole test system, define the execution flow, collect results and communicate them by using reports or logging to a database. Use the application development tools to create the particular tests you will want to perform on the unit under test. Measurement and control services provide an interface for software to control the instruments and hardware that are part of the test system.

3

Choosing the Right Software Application Development Environment

Application development environments (ADEs) play a critical, visible role in a test software framework. With these tools, the system developer designs and integrates the system that takes measurements, displays information to the end user, connects with other applications, and much more. Due to the ever increasing role of software in test system implementation, system developers will spend most of their development time working with an ADE. It is critical to select an ADE that not only is easy to use, but also can support multiple platforms and integrate easily with measurement and control services such as drivers. Other features that should be considered when selecting an ADE for the development of your test system are its presentation and reporting features, how protected you are from the obsolescence of the product, and what kind of training and support is available worldwide. This paper discusses how three different ADEs – NI LabVIEW, NI LabWindows/CVI and Microsoft Visual Studio .Net – compare on these characteristics.

Factors to Consider When Selecting an ADE

		Application Development Environment		
		NI LabVIEW	NI LabWindows/CVI	Microsoft Visual Studio
Factor	Easy to Use	●	◗	◗
	Measurement and Analysis Capabilities	●	●	○
	Integration with Measurement and Control Drivers	●	●	○
	Training and Support	●	●	◗
	Platform Independence	●	◗	◗
	Presentation and Reporting Features	●	◗	○
	Protection Against Obsolescence	●	●	◗

Good ○, Better ◗, Best ●

Figure 3-1. Different ADEs provide different benefits and challenges when used to develop your test system.

Ease of Use

Because the ADE is where the heart of an automated system is developed, ease of use in these tools is critical to the productivity of software engineer. Ease of use goes beyond how quickly someone can get up and running. With easy-to-use ADEs, developers can easily integrate processing routines with multiple measurement devices, create sophisticated user interfaces, deploy and maintain an

application, and modify the application as product designs evolve and system needs expand. Other features that should be included with the ADE include extensive documentation and example code.

Measurement and Analysis Capabilities

It is critical that the ADE used to develop a test system can seamlessly manage and process measurements. To do this effectively, the ADE should incorporate measurement data types directly in the environment so that the data is easy to use in additional processing routines. For maximum development productivity, the ADE should include comprehensive statistical and numerical analysis functions, as well as high-performance signal processing and control algorithms common in measurement applications.

Integration with Measurement and Control drivers

Too often, developers of test systems assume that the existence of a device driver alone is sufficient for effectively integrating their measurement device. The existence of a driver is not enough; measurement and control drivers should be integrated as seamlessly as possible with the ADE. In the ideal case, the software that controls the measurement devices is transparent, appearing only as part of the ADE. This ideal implementation guarantees maximum flexibility in development and a scalable architecture that organizations can deploy on all of the platforms targeted by the ADE.

Training and Support

The ease of use of an ADE can go so only far in making it simple for new users to learn the application. Hence, the ADE vendor should provide manuals and online training so engineers can quickly learn how to use their product. Advanced users might also need classroom training to further their knowledge and learn more about system level design concepts. This classroom training should give developers the opportunity to attain proof of their knowledge by going through a certification process. Another consideration when selecting an ADE is the vendor support accessible to when developing your application, such as phone and e-mail support. Furthermore, if you are going to standardize on an ADE worldwide, you will want to consider whether your engineers around the world will have access to support in their own language.

Platform Independence

Test software applications today target many different architectures. It is important that whatever ADE you choose can be flexible enough to support all these different architectures as seamlessly as possible. Different OSs such as Windows, Linux® and Mac OS X can offer different benefits depending on the application. Engineers should be able to port their code from one platform to the other. If your ADE is not compatible with these other platforms, you will need to use different ADEs for different projects and spend valuable time porting your existing intellectual property from one platform to the other.

Presentation and Reporting Features

Test applications present many challenges in the area of presentation and reporting because of their emphasis on the graphical representation of data. The ADE should have multiple visual components for data visualization such as charts, graphs, knobs, and meters. Reporting should also be easy in order to facilitate the communication of the information acquired by the system. Some of the most popular reports, such as MS Word and MS Excel, should be easy to generate. The communication of results should also be easy to implement by either publishing the application on the Web or logging information to a database.

Protection against Obsolescence

Standardizing on an ADE for the development of your test system is a big commitment. It is important that your investment is protected from the obsolescence of the product. One of the characteristics you should consider is the product track record of integrating with the latest software technologies and its commitment to protecting you against discontinuous shifts in test software development. Furthermore, the product should offer routine upgrades to add new functionality.

LabVIEW Graphical Programming Environment

LabVIEW is a graphical development language that helps engineers and scientists create flexible, scalable test applications rapidly and at minimal cost. LabVIEW uses a graphical development paradigm instead of relying on text based programming as implemented by other programming languages such as Visual Basic, C++ and C#. The LabVIEW graphical dataflow language and block diagram approach naturally represent the flow of your data and intuitively map user interface controls to your data, so you can easily view and modify your data or control inputs. Figure 3-1 depicts the block diagram of an application and its respective front panel written in LabVIEW.

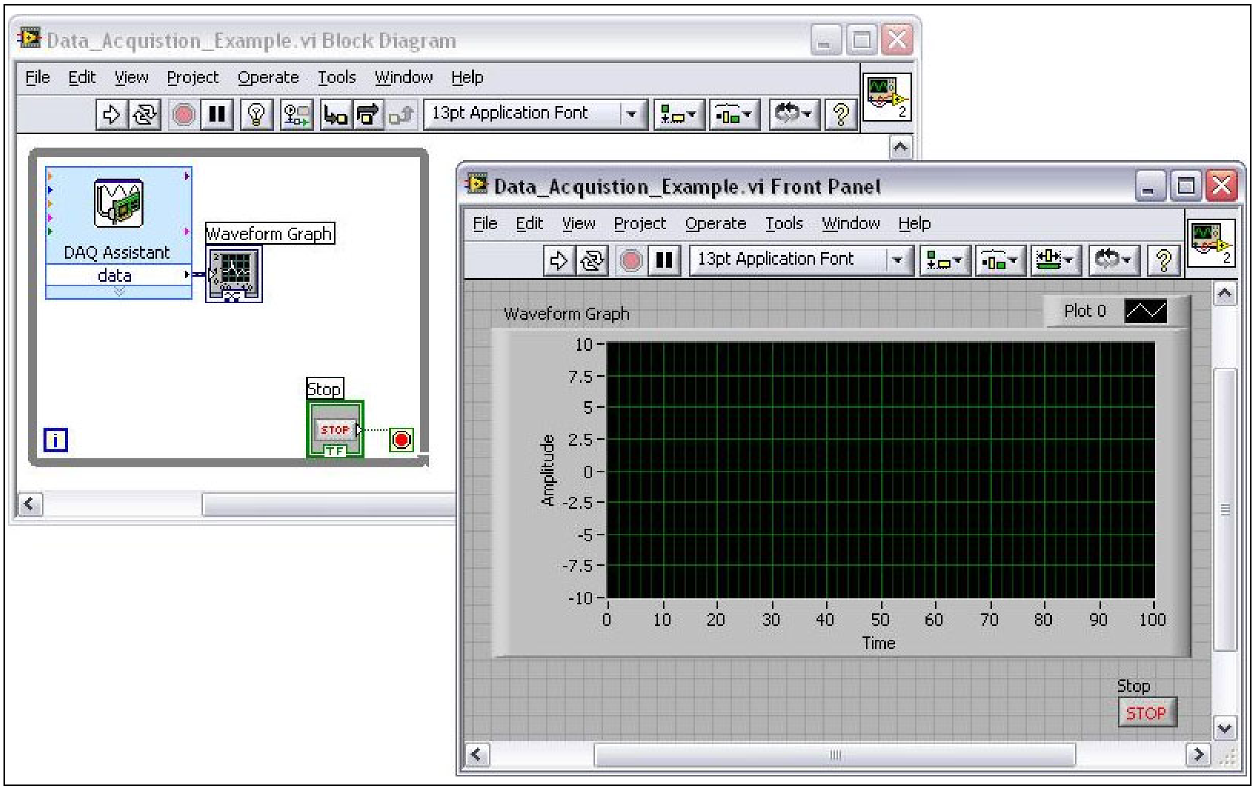

Figure 3-2. The LabVIEW ADE can quickly and clearly development test applications.

LabVIEW also includes features to facilitate access to the extensive documentation included with the product. With the Context Help feature you and take advantage of the graphical nature of LabVIEW to access subVI documentation by simply hovering over the subVI with the cursor. LabVIEW also emphasizes the use of the hundreds of example programs available with the product and online as a means of demonstrating and teaching different features.

Despite the sophistication of the underlying algorithms, LabVIEW analysis tools are easy to use. More than 15 Express VIs for data analysis, such as the Spectral Measurements Express VI, reduce the complexity of implementing analysis in your application through interactive configuration dialogs in which you can preview analysis results immediately.

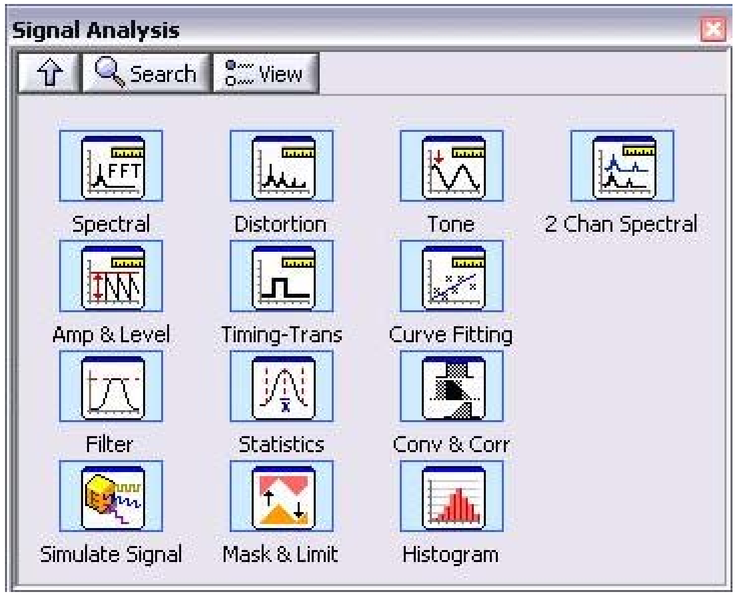

Figure 3-3. Signal analysis Express VIs are powerful, easy-to-use programming tools for analysis in your application.

These and other measurement analysis tools can input real-world, time-domain signals directly from data acquisition hardware and provide results ready for charting, graphing, or further processing. With these functions, you easily can determine signal characteristics such as DC/rms levels, total harmonic distortion (THD/SINAD), impulse response, frequency response, and cross-power spectrum.

One of the biggest strengths of LabVIEW is its tight integration with measurement and control drivers. LabVIEW simplifies connecting to and communicating with thousands of instruments from hundreds of vendors. With LabVIEW, you can quickly acquire data from GPIB, serial, Ethernet, PXI, USB, and VXI instruments using instrument drivers, interactive assistants, and built-in instrument I/O libraries. Furthermore, LabVIEW includes easy-to-use libraries and interactive assistants to communicate with NI modular instruments and data acquisition products.

National Instruments offers LabVIEW training for any level of expertise. While basic courses are targeted towards nonprogrammers and existing developers who want to learn the product, intermediate and advanced users will also find content that is useful to their level of expertise. Onsite courses help organizations train a large number of developers quickly without having to leave the company. Online and self-paced courses target those developers who wish to increase their knowledge on their own time and at their own pace. The extensive training opportunities for LabVIEW are complemented by the worldwide support available to users. National Instruments application engineers provide support for LabVIEW from local branches around the world. This means that engineers will be able to receive phone, e-mail, or forum support for LabVIEW in their language wherever they are located.

Although LabVIEW is usually seen as a MS Windows application, the original product ran on the Macintosh. National Instruments ported LabVIEW to Windows as it began to dominate the desktop PC industry. The LabVIEW commitment to new platforms continues today. LabVIEW continues to run on both Windows and Mac OS X but now also runs on Linux because of its increasing popularity among customers. Being able to run LabVIEW VIs on different OSs means that no matter which computing platform you need to work with, you will be able to use your NI LabVIEW knowledge. LabVIEW can even run on other targets such as real-time systems, FPGAs, and DSPs.

The presentation and reporting features of LabVIEW are a big part of why the ADE is so well suited for test software development. LabVIEW contains multiple graphs, charts, meters, knobs, and switches both in 2D and 3D in order to facilitate the representation of measurement data graphically. The ADE also includes the LabVIEW Report Generation Toolkit, which facilitates the creation of reports in MS Word and Excel format. If it is necessary to communicate results by exporting the application through the Web, LabVIEW Remote Panels can be used to display the front panel over the Web on any browser. On the other hand, if the results of your measurements need to be logged to a database, the LabVIEW Database Connectivity Toolkit provides a set of easy-to-use tools with which you can quickly connect to local and remote databases and perform many common database operations.

Finally, National Instruments has emphasized throughout time that it is committed to help its LabVIEW customers fight obsolescence. Even though a large amount of development effort has been focused on adding new features and integrating new technologies, running code from previous versions on newer version has always been a priority. Running older code on newer versions of the product means that the valuable resources that were dedicated to creating previous applications will not be wasted and can bridge the gap to newer development

LabWindows/CVI, an ANSI C Development Environment

LabWindows/CVI is a proven test and measurement ANSI C development environment that greatly increases the productivity of engineers and scientists. Figure 3-4 displays the LabWindows/CVI development environment.

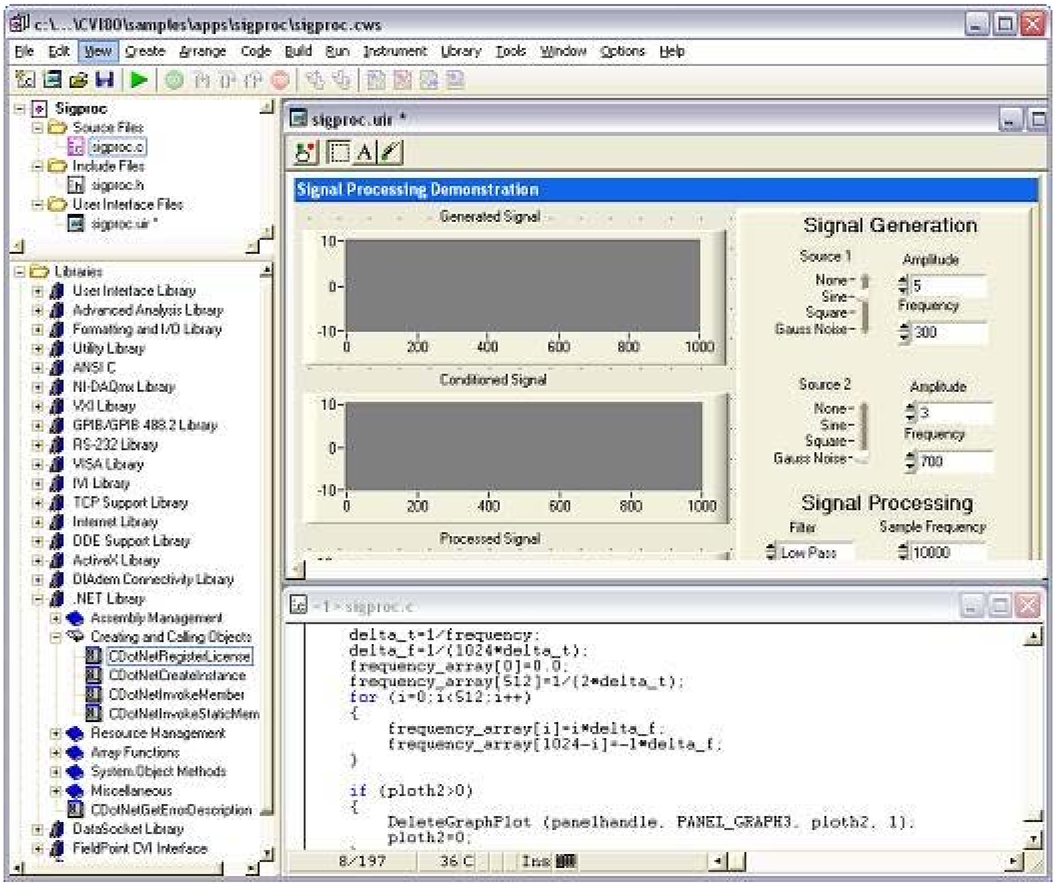

Figure 3-4. LabWindows/CVI 8.0 features a complete workspace you can use to quickly develop, debug, and manage large applications.

Engineers and scientists use LabWindows/CVI to develop high-performance, stable applications in the manufacturing test, military and aerospace, telecommunications, design validation, and automotive industries. LabWindows/CVI streamlines development in these areas with hardware configuration assistants, comprehensive debugging tools, and interactive execution capabilities that developers can use to run functions at design time.

Toolkits such as the Advanced Analysis Library complement the analysis libraries included with LabWindows/CVI to help engineers make analyze their measurement data. The LabWindows/CVI Advanced Analysis Library offers a comprehensive set of functions for analyzing your data. With these powerful analysis routines, you can easily convert raw data into useful information and build test applications. The Advanced Analysis Library includes functions for signal

generation, one-dimensional 1D and 2D array manipulation, complex operations, signal processing, statistics, and curve fitting.

LabWindows/CVI is an industry leader in instrument control and connectivity, through an Instrument Driver Network of more than 4,000 instrument drivers from more than 200 vendors. You can use these drivers to easily program instrument control applications. With Instrument I/O Assistant, you can generate code to communicate with devices such as serial, Ethernet, and GPIB instruments without using an instrument driver. Instrument I/O Assistant offers a simple interface for quickly prototyping applications and autoparsing instrument data without any programming. You can easily import the code generated into any existing application, which removes the tedium of writing instrument connectivity, basic communication, and string parsing code. In addition to the integrated NI-DAQmx Libraries, LabWindows/CVI also provides DAQ Assistant, an interactive interface to the data acquisition driver framework.

The training and support structure available for LabVIEW, is also present for LabWindows/CVI. LabWindows/CVI has training courses that target different levels of expertise with the product. Onsite courses are also available for organizations who need to train a large number of developers quickly without having to leave the company. Options are also available for engineers who wish to increase their knowledge on their own time and at their own pace in the form of online and self-paced courses. To complement the training opportunities for LabWindows/CVI, worldwide support is provided by National Instruments application engineers from local branches around the world.

By maintaining the backward compatibility of LabWindows/CVI, National Instruments helps to protect you from obsolescence. Not only can you run C code developed many years ago, or LabWindows/CVI code created in previous version of the product, but you can also have the applications run faster with new optimizing compiler integration. The LabWindows/CVI commitment to backward compatibility is critical to industries that value longevity and continuity such as military and aerospace.

Microsoft Visual Studio .Net (C++, Visual Basic .Net, C#, and ASP.NET)

Visual Studio .NET provides a very powerful ADE by supporting four programming languages – C++, Visual Basic .Net, C# and ASP.NET. The option to select any of these programming languages means that you can use one tool while taking advantage the expertise of your developers even if their knowledge focuses on different programming languages. Applications developed in Visual Studio .NET can be run on a PC as well as through the Web by using the ASP.NET language.

The Visual Studio .Net provides functionality to develop in different programming languages such as C++, Visual Basic .Net and C#. By enabling these programming languages to compile to the Common Language Runtime you can add libraries

developed in different languages. On the other hand, the fact that the .Net platform works only with Microsoft Windows means that you are very limited in the number of OSs that can run your application. Furthermore, porting your application to another OS in the future might require rewriting the application in a different language.

By default, Visual Studio .NET does not include any functionality to integrate with measurement and control drivers or perform any analysis operations. Components, such as those offered by NI Measurement Studio, can provide access to measurement and instrument drivers and analysis functionality. These components increase the integration of the ADE with instrument and measurement drivers by providing interactive assistants to generate code automatically. In contrast, there are certain features of the .NET framework that make it inherently difficult to communicate with certain instruments. The .NET framework executes code in the Common Language Runtime, which isolates you from accessing the hardware. Unable to access the hardware, it is very difficult to write directly to instrument registers. In order to do this, one would have to create a DLL and then call it from a .NET application.

Visual Studio .NET offers few presentation and reporting capabilities by default. Out of the box, the ADE provides enough features to generate a standard Windows application by offering text boxes, combo boxes, list boxes, buttons and other components that are needed to create a basic application. In order to use more powerful components to display data such as graphs and charts, you will need to purchase a set of components for this particular application. This problem is also repeated in the lack of reporting tools for any of the programming languages in Visual Studio .NET. On the other hand, the .NET framework includes powerful features for reporting by storing information to a database. ADO .NET, a rich library of database functionality, can be used to communicate with and perform operations on many different databases.

Because the focus of .NET lies in business, IT, and Web-based applications instead of automated test, guaranteeing the longevity of the programming language and avoiding discontinuous shifts is not a priority. Applications that focus on IT can have a life cycle of a few months versus years in the case of automated test. For example, even though it is possible to integrate DLLs into the .NET, it requires the developer to manually invoke the function and guarantee that the DLL data types match those in .NET. At first this might not seem very challenging, but if you need to communicate with hundreds of functions of an instrument driver, this process can be very time-consuming. On the other hand, incorporating your existing ActiveX components into a .NET automated test application is easier than incorporating DLLs. Visual Studio .NET can generate wrappers around your ActiveX components to expose them as .NET objects.

4

Hybrid Systems – Integrating Your Multivendor, Multiplatform Test Equipment

The Challenge for Test Systems Developers

In recent years, a number of test and measurement platforms and communication buses have emerged as options in test equipment. This proliferation of instrumentation options offers both an opportunity and a challenge to system developers. Engineers now have more flexibility in customizing a system with a wide selection of test equipment, but they face the challenge of integrating instruments built upon distinct platforms into a single system.

You have the choice of modular and stand-alone hardware as well as a variety of communication buses, including USB, Ethernet/LAN, and PCI Express. You also face the choice between the software flexibility provided by virtual instruments – where the user defines the instrument's functionality – and the fixed functionality provided by vendor-defined instruments. Because budget and time constraints often require developers to reuse some test equipment and because mid-life upgrades and system repairs may require integration of new test equipment, most systems will contain a mix of several of these bus and platform technologies.

This paper discusses how to integrate multivendor, multiplatform test equipment into a single hybrid system, which both protects your existing investment through reuse of existing hardware and software and simplifies the task of upgrading and maintaining the system.

What Is a Hybrid System?

Hybrid systems combine test and measurement components from modular instrumentation platforms such as PXI and VXI and stand-alone instruments that connect externally across GPIB, USB, and Ethernet/LAN. The hybrid system diagram of Figure 4-1 gives one example of a hybrid topology, but any number of combinations is possible. In the diagram, the nerve center of the system is a PC – either a standard PC or an embedded PXI controller. The instruments themselves include PXI and PCI plug-in devices as well as traditional instruments connecting to the PC over GPIB, USB and Ethernet/LAN/LXI.

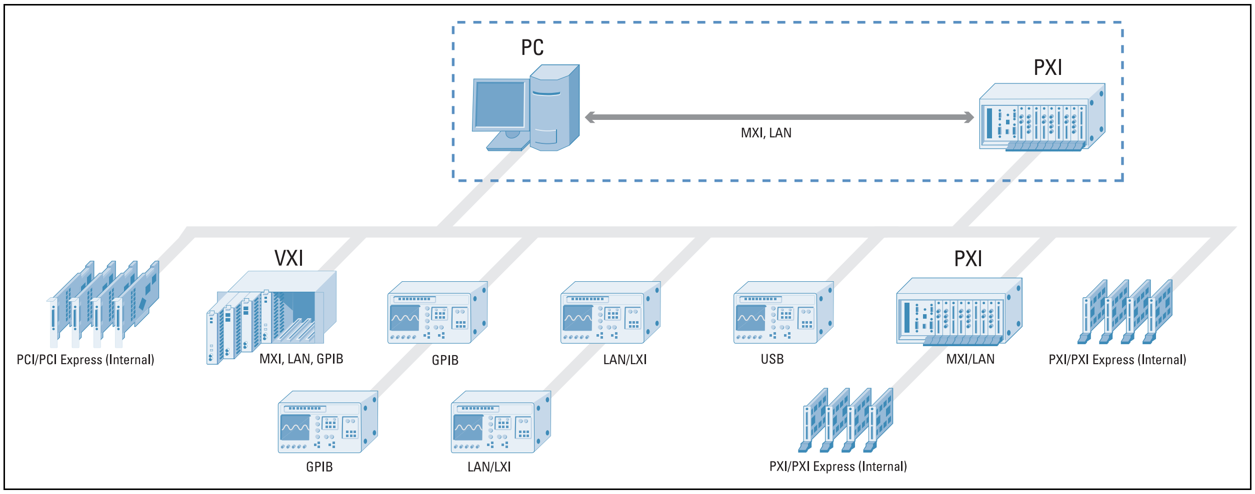

Figure 4-1. One Possible Hybrid System Topology

The key to creating and maintaining a hybrid system is implementing a system architecture that transparently accommodates multiple bus technologies and uses an open, multivendor hardware platform to achieve I/O connectivity. With the proper computer platform and driver, application, and test system management software, you can combine the strengths of many types of instruments, including legacy equipment and specialized devices. It is important to fully recognize that no single platform or bus technology meets all needs, though each has unique strengths. For instance, GPIB has the largest installed base of compatible instruments, and is therefore useful for equipment reuse and specialized equipment. PCI and PCI Express offer the best bandwidth and latency performance, critical in applications where large amounts of data are being generated or acquired. PXI and PXI Express offer the same bandwidth and latency specifications as PCI and PCI Express but add the industry's best timing, triggering, and synchronization capabilities. The greatest strength of USB lies in its autodetecting plug-and-play connectivity. And Ethernet/LAN is often the best choice for distributed or remote systems; the LXI specification adds the ability to synchronize Ethernet/LAN instruments.

In a hybrid system, it is the PC and software that pull together the various instrumentation hardware pieces into a single system. The next section outlines in detail how with a properly planned system architecture, you can simplify your hardware integration challenge by:

- Incorporating existing/legacy test routines into the new system with minimal rework
- Introducing future test routines into the system without a complete system redesign
- Simplifying replacement of individual instruments and I/O devices.

Making the Right Software Decisions

Modular hardware and software is the best way to integrate your multivendor, multiplatform test equipment into a single hybrid system. Though some vendors may offer vertical software solutions for specific instruments, the most useful system architecture is one that breaks up the hardware and software functions into interchangeable modular layers so that your system is neither tied to a particular piece of hardware or to a particular vendor. Figure 4-2 shows a layered test system architecture. This layered approach provides the best code reuse, modularity, and longevity. The remainder of this section will discuss each layer in turn.

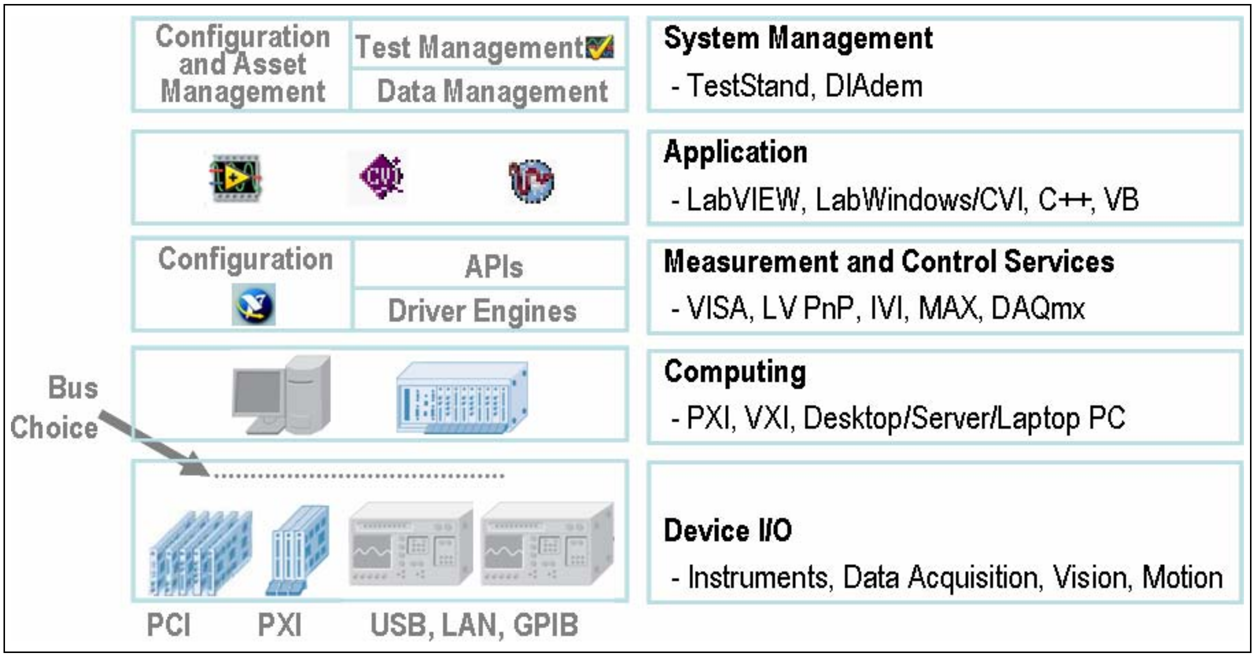

Figure 4-2. Test System Architecture

At the lowest level is the device I/O layer. This is the hardware layer which contains the individual instruments, signal conditioning, and fixturing. Hardware may be stand-alone or modular. The instruments may rely on software-defined functionality, such as with virtual/synthetic instruments, or they may be of a vendor-defined fixed functionality. They may communicate on various instrumentation buses including PCI, PCI Express, GPIB, Ethernet/LAN/LXI, and USB.

The computing layer includes the computer used to control modular instrumentation and connect to stand-alone buses. This is the core or nerve center of the hybrid system. It may be a rugged embedded controller in the case of PXI or VXI. It may be a server or industrial PC. Or it may simply be a laptop or desktop PC. The choice is dependent on the application needs. In the case of PCs and PXI systems, the computing layer components run standard OSs, such as Windows and Linux®, and share the same standard PC backplane technologies, so they are easily interchangeable.

A very important but somewhat invisible part of the test software architecture required to accommodate multiple platforms in hybrid systems is the measurement and control services Layer. This layer serves to provide configuration support and hardware abstraction. It contains the instrument drivers that bridge the hardware to the software. A few examples include Virtual Instrument Software Architecture (VISA), IVI, and Measurement & Automation Explorer (MAX). VISA is a vendor-neutral software standard for configuring, programming, and troubleshooting instrumentation systems comprising GPIB, VXI, serial (RS232/485), Ethernet, USB and/or IEEE 1394 interfaces. It is a useful tool since the API for programming VISA functions is similar for a variety of communication interfaces. Interchangeable virtual instruments (IVI) is a standard for instrument drivers, useful for instrument replacement because with it you can interchange instruments in a system with minimal test software modification, for specified classes of instruments such as oscilloscopes and switches. An application that uses IVI can substitute one instrument for another of the same IVI class, regardless of manufacturer or bus connection. MAX is an NI utility that simplifies configuring your various test hardware and software from one integrated interface. It also provides a means of importing and exporting channel lists and configurations. The right measurement and control services software can simplify the replacement of individual instruments and I/O devices in a hybrid system, because you can use it to make your test code hardware-independent.

The application layer is composed of the individual test programs like a DC voltage measurement or a frequency power spectrum. Though applications can be written in just about any software language, there are several programming environments that include special tools for interfacing to test and measurement equipment, such as National Instruments LabVIEW and LabWindows/CVI (an ANSI C environment). These environments are ideal for open connectivity, with built-in functions in VISA, IVI, Ethernet/LAN/LXI, and USB communication. They also have a suite of analysis and data visualization tools. You can incorporate existing/legacy test routines into the new system by calling the existing code from your new application.

At the highest level, the system management layer provides a framework to sequence the test routines as well as log data, generate reports, and manage user privileges. NI TestStand is the industry de facto standard for this kind of system management software. Regardless of whether individual tests are coded in LabVIEW, LabWindows/CVI, Visual Basic, C, C++, or a .NET programming language, NI TestStand can help to quickly build test sequences and then manage the test execution, including resource scheduling and reporting. By organizing individual test routines through a full-featured test management environment such as NI TestStand, you can introduce future test routines into the system without a complete system redesign.

The layered test system architecture outlined above provides useful way to think about the hardware and software architecture of a hybrid test and measurement system. It is important to remember that without a flexible, modular software model, many of the potential gains of a hybrid system cannot be realized. The final section builds upon the test system architecture concept and offers a few additional implementation tips for designing a hybrid system.

Designing Your System

When designing any test and measurement system, the measurement needs of the specific unit or process under test determine the required functionality of the test equipment. Once these needs are assessed, the developer can begin to select equipment. In a hybrid system, it is common to reuse any appropriate equipment already available, and then add instruments that fill the remaining gaps. Some instruments may be highly specialized and offered only by a particular vendor; but most instruments, such as switches, DMMs, function generators, and digitizers/scopes, are available in a variety of forms (stand-alone and modular) and from a variety of vendors. Choose the instrument that makes sense for you, keeping in mind future needs for extensibility, interchangeability, and upgrade.

Even a single instrument may have multiple connectivity options. Several vendors offer boxed instruments that may include two or more of USB, GPIB, Serial, and LAN/Ethernet connections. When choosing the instrument and interface, prepare not only to meet the basic measurement requirements of the unit under test, but also to have the ability to adapt to growing system measurement needs. This means going beyond features such as resolution, frequency, sampling rate, and channel count, and looking for complete measurement functionality. With virtual instrumentation, you can create a user-defined system that can easily adjust to changing measurement needs without having to replace hardware.

It is also important to think about timing and synchronization requirements. The computing center may need to coordinate tasks between instruments, or instruments may need to communicate directly with each other in a way that requires hardware synchronization. Examples of common timing and synchronization tasks include handshaking between a DMM and switch, phase-lock-looping a waveform generator with a digitizer, or synchronizing an RF downconverter with an IF digitizer. One advantage of choosing the PXI platform as the core of a hybrid system is the integrated timing and triggering capabilities that PXI offers. Unlike stand-alone instruments, the moment you plug in PXI modular instruments to the PXI chassis, all timing and triggering lines are instantly connected and ready for use, including access to a highly stable 10 or 100 MHz system clock, and trigger lines offering nanosecond skew. Plus, PXI systems offer advanced options like IRIG A, IRIG B, and GPS. Largely for these benefits, users choose PXI modular instruments for parts of the hybrid system that require the tightest synchronization and connect to stand-alone instruments as peripherals for specialized needs or equipment reuse.

After choosing the instruments, buses, and hardware platforms, the task remains to create the application software. Because test software development usually costs more (because of developer time) than the system hardware, the right software pieces are critical for staying on time and on budget. To save time and money now and in the future, develop the test system software using layered test system architecture. Keep the driver, application, and test management layers fully separated. Also spend a little effort to select the appropriate drivers, for instance, choosing between IVI for maximum reuse and portability, plug-and-play drivers

for a compromise between portability and low-level control, and direct I/O. Direct I/O offers the lowest-level control, but at the cost of intensive programming and no interchangeability.

Conclusion: Building Hybrid Systems

With hybrid instrumentation systems, developers can integrate flexible modular instrumentation platforms with cabled stand-alone instruments to construct a system that takes advantage of the best features of both approaches. The key to successfully integrating PXI and VXI modular instruments with traditional instruments cabled across GPIB, USB, or LAN/Ethernet is the layered test system architecture that abstracts away instrument and vendor-specific concerns.

5

Instrument Bus Performance – Making Sense of Competing Bus Technologies for Instrument Control

In 1997 Hewlett-Packard (now Agilent) strongly claimed that IEEE 1394 was ideally situated to be the new leading bus technology in instrument control. HP advocated abandoning the then-leading technology, GPIB, in light of IEEE 1394 potential. In the decade since, IEEE 1394 has failed to gain anything more than a marginal adoption in instrumentation outside of imaging, but some test and measurement companies have continued to try to identify a single instrument control bus to replace all others.

While other bus technologies have certainly proved more successful than IEEE 1394 in fulfilling a broad range of application needs, even GPIB, the most adopted instrument control standard in the past 40 years, cannot claim to be categorically superior to all other buses. Today, USB, PCI Express, and Ethernet/LAN have gained attention as attractive communication options for instrument control. Some test and measurement vendors and industry pundits have claimed that one of these buses, by itself, represents a solution for all instrumentation needs. In reality, it is most likely that two or more bus technologies will continue to coexist in future test and measurement systems because each bus has its own strengths. The challenge for today's test engineer is not to choose a single bus or platform on which to standardize every single application, but to choose a bus or platform appropriate for a specific application or even a specific part of an application.

This paper presents a head-to-head comparison of the most popular instrumentation buses, so that test engineers can make informed decisions when choosing the bus and platform technologies to meet their application-specific needs. Specific bus technologies discussed below include GPIB, USB, PCI, PCI Express, and Ethernet/LAN/LXI.

Understanding Bus Performance

First, it is important to outline the relevant performance criteria for instrument control buses, in order to set a baseline for evaluation and comparison.

Bandwidth

When considering the technical merits of alternative buses, bandwidth and latency are two of the most important bus characteristics. Bandwidth measures the rate at which data is sent across the bus, typically in MB/s (106 bytes per second). A bus with high bandwidth is able to transmit more data in a given period than a bus with low bandwidth. Most users recognize the importance of bandwidth because it

affects whether their data can be sent across the bus to or from a shared host processor as fast as it is acquired or generated and how much onboard memory their instruments will need. Bandwidth is important in applications such as complex waveform generation and acquisition as well as RF and communications applications. High-speed data transfer is particularly important for virtual and synthetic instrumentation architectures. The functionality and personality of a virtual or synthetic instrument is defined by software; in most cases, this means data must be moved to a host PC for processing and analysis. Figure 5-1 charts the bandwidth (and latency) of all the instrumentation buses examined in this paper.

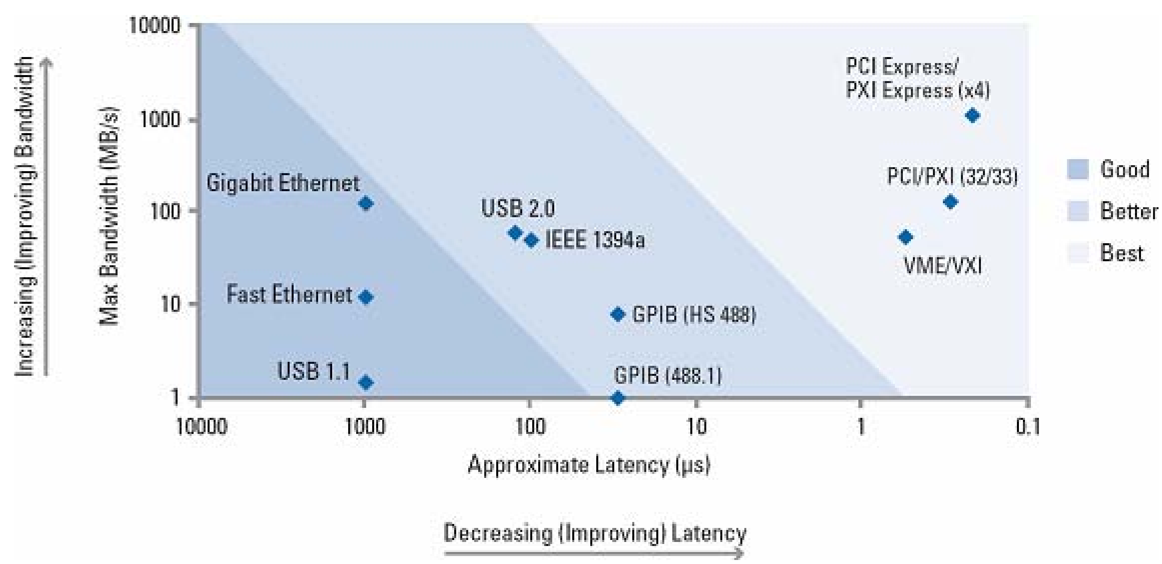

Figure 5-1. Bandwidth versus Latency for Instrumentation Buses

Latency

Latency measures the delay in data transmission across the bus. By analogy, if we were to compare an instrumentation bus to a highway, bandwidth would correspond to the number of lanes and the speed of travel, while latency would correspond to the delay introduced at the on and off-ramps. A bus with low (meaning good) latency would introduce less delay between the time data was transmitted on one end and processed on the other end. Latency, while less observable than bandwidth, has a direct impact on applications where a quick succession of short, choppy commands are sent across the bus, such as in handshaking between a digital multimeter (DMM) and switch, and in instrument configuration.

Message versus Register-Based Communication

Buses that use message-based communication are generally slower because this mode of communication adds overhead in the form of command interpretation and padding around the data. With register-based communication, data transfer occurs by directly writing and reading binary data to and from hardware registers on the device, resulting in a faster transfer. Register-based communication protocols are

most common to internal PC buses, where interconnects are physically shorter and the highest throughput is required. Message-based communication protocols are useful for transmitting data over longer distances and where higher overhead costs are acceptable. It should be noted that the latency and bandwidth metrics are partially dependent whether the bus uses message or register-based communication and so this parameter is partially captured in those metrics.

Long-Range Performance

For remote monitoring applications and for systems that involve measurement over a large geographical area, range becomes important. Performance in this category can be viewed as a tradeoff with latency, because the error checking and message padding added to overcome physical limitations of sending data over longer cables can add delays to sending and receiving the data.

Instrument Setup and Software Performance

Ease of use in terms of instrument setup and software performance is the most subjective criterion examined here. Nonetheless, it is important to discuss. Instrument setup describes the "out of the box" experience and setup time. Software performance relates to how easily users can find interactive utilities or standard programming APIs such as VISA to communicate with and control the instrument.

Robustness of Connector

The physical connector for the bus affects whether it is suitable for industrial applications and whether additional effort will be required to "ruggedize" the connection between the instrument and the system controller. Figure 5-2 presents photos of several instrumentation bus connectors.

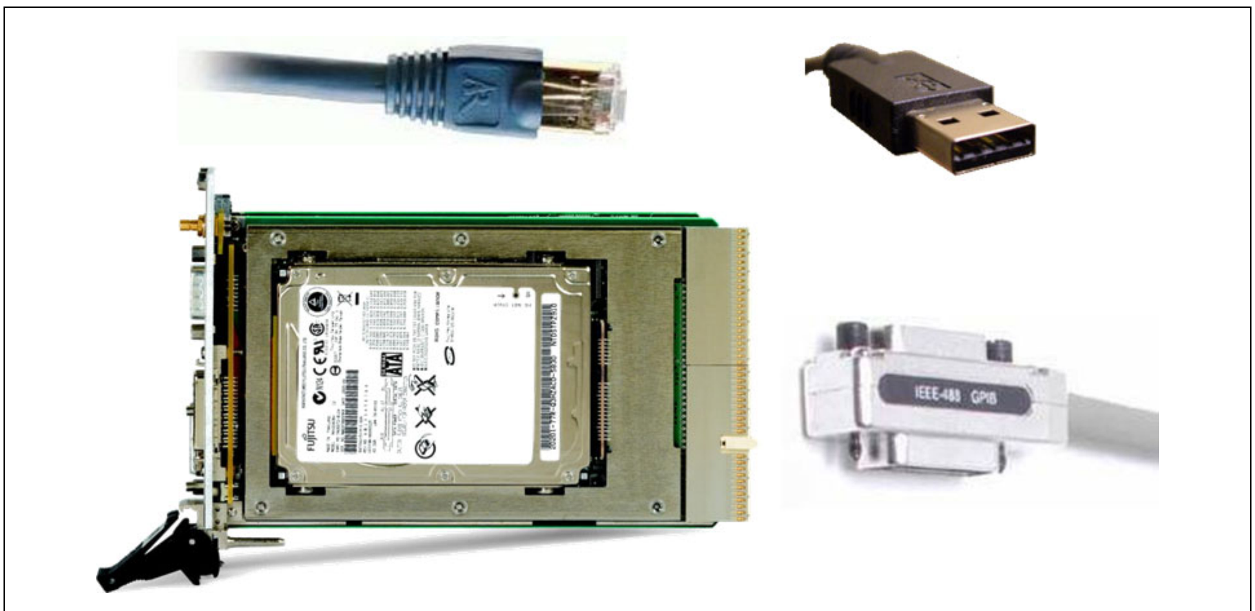

Figure 5-2. Connectors for Ethernet, USB, PXI, and GPIB (not to scale). The connector for PXI is an integrated part of the modular instrument on which it resides.

Buses

GPIB

The first bus we will look at is the IEEE 488 bus, familiarly known as GPIB (general-purpose interface bus). GPIB is a proven bus designed specifically for instrument control applications. GPIB has been a robust, reliable communication bus for 30 years and is still the most popular choice for instrument control because of its low latency and acceptable bandwidth. It currently enjoys the widest industry adoption, with a base of more than 10,000 instrument models with GPIB connectivity.

With a maximum bandwidth of about 1.8 MB/s, it is best suited for communicating with and controlling stand-alone instruments. The more recent, high-speed revision, HS488, increased bandwidth up to 8 MB/s. Transfers are message-based, often in the form of ASCII characters. Multiple GPIB instruments can be cabled together to a total distance of 20 m, and bandwidth is shared among all instruments on the bus. Despite relatively lower bandwidth, GPIB latency is significantly lower (better) than that of USB and especially Ethernet. GPIB instruments do not autodetect or autoconfigure when connected to the system; though GPIB software is among the best available, and the rugged cable and connector are suitable for the most demanding physical environments. GPIB is ideal for automating existing equipment or for systems requiring highly specialized instruments.

USB

USB (universal serial bus) has become popular in recent years for connecting computer peripherals. That popularity has spilled over into test and measurement, with an increasing number of instrument vendors adding USB device controller capabilities to their instruments.

Hi-Speed USB has a maximum transfer rate of 60 MB/s, making it an attractive alternative for instrument connectivity and control of stand-alone and some virtual instruments with data rates below 1 MS/s. Though most laptops, desktops, and servers may have several USB ports, those ports usually all connect to the same host controller, so the USB bandwidth is shared among all the ports. Latency for USB falls into the better category (between Ethernet at the slow end and PCI and PCI Express at the fast end), and cable length is limited to 5 m. USB devices benefit from autodetection, which means that unlike other technologies such as LAN or GPIB, USB devices are immediately recognized and configured by the PC when a user connects them. USB connectors are the least robust and least secure of the buses examined here. External cable ties may be needed to keep them in place.

USB devices are well-suited for applications with portable measurements, laptop or desktop data logging, and in-vehicle data acquisition. The bus has become a popular communication choice for stand-alone instruments due to its ubiquity on PCs and especially due to its plug-and-play ease of use. The USB Test and Measurement Class (USBTMC) specification addresses the communication requirements of a broad range of test and measurement devices.

PCI

PCI and PCI Express achieve the best bandwidth and latency specifications among all the instrumentation buses examined here. PCI bandwidth is 132 MB/s, with that bandwidth shared across all devices on the bus. PCI latency performance is outstanding; benchmarked at 700 ns, compared to 1 ms in Ethernet. PCI uses register-based communication. Unlike the other buses mentioned here, PCI does not cable to external instruments. Instead, it is an internal PC bus used for PC plug-in cards and in modular instrumentation systems such as PXI, so distance measures do not directly apply. Nonetheless, the PCI bus can be "extended" by up to 200 m by the use of NI fiber-optic MXI interfaces when connecting to a PXI system. Because the PCI connection is internal to the computer, it is probably fair to characterize the connector robustness as being constrained by the stability and ruggedness of the PC in which it resides. PXI modular instrumentation systems, which are built around PCI signaling, enhance this connectivity with a high-performance backplane connector and multiple screw terminals to keep connections in place. Once booted with PCI or PXI modules in place, Windows automatically detects and installs the drivers for modules.

An advantage that PCI (and PCI Express) share with Ethernet and USB is that they are universally available in PCs. PCI is one of the most widely adopted standards in the history of the PC industry. Today, every desktop PC has either PCI slots, PCI Express slots, or both. In general, PCI instruments can achieve lower costs, because these instruments rely on the power source, processor, display, and memory of the PC that hosts them, rather than incorporating that hardware in the instrument itself.

PCI Express

PCI Express is similar to PCI. It is the latest evolution of the PCI standard, much as Hi-Speed USB is to USB. Therefore, much of the above evaluation of PCI applies to PCI Express as well. The main difference between PCI and PCI Express performance is that PCI Express is a higher bandwidth bus and gives dedicated bandwidth to each device. Of all the buses covered in this tutorial, only PCI express offers dedicated bandwidth to each peripheral on the bus. GPIB, USB, and LAN, divide bandwidth across the connected peripherals. Data is transmitted across point-to-point connections called lanes at 250 MB/s per direction. Each PCI Express link can be composed of a multiple lanes, so the bandwidth of the PCI Express bus depends on how it is implemented in the slot and device. A x1 (by 1) link provides 250 MB/s; a x4 link provides 1 GB/s; and a x16 link provides 4 GB/s dedicated bandwidth. It is important to note that PCI Express achieves software backward compatibility, meaning that users moving to the PCI Express standard can preserve their software investments in PCI. PCI Express also extensible by external cabling.

High-speed, internal PC buses were designed for rapid communication. Consequently PCI and PCI Express are ideal bus choices for high-performance, data-intensive systems where large bandwidth is required, and for integrating and synchronizing several types of instruments.

Ethernet/LAN/LXI

Ethernet has long been an instrument control option. It is a mature bus technology and has been widely used in many application areas outside of test and measurement. 100BaseT Ethernet has a theoretical max bandwidth of 12.5 MB/s. Gigabit Ethernet, or 1000BaseT, increases the max bandwidth to 125 MB/s. In all cases, Ethernet bandwidth is shared across the network. At 125 MB/s Gigabit Ethernet is theoretically faster than Hi-Speed USB, but this performance quickly declines when multiple instruments and other devices are sharing network bandwidth. Communication along the bus is message based, with communication packets adding significant overhead to data transmission. For this reason, Ethernet has the worst latency of the bus technologies featured in this tutorial.

Nonetheless, Ethernet remains a powerful option for creating a network of distributed systems. It can operate at distances up to 85 to 100 m without repeaters and with repeaters has no distance limits. No other bus has this range of separation from the controlling PC or platform. As with GPIB, autoconfiguration is not available on Ethernet/LAN. Users must manually assign an IP address and subnet configuration to their instrument. Like USB and PCI, Ethernet/LAN connections are ubiquitous in modern PCs. This makes Ethernet ideal for distributed systems and remote monitoring. It is often used in conjunction with other bus and platform technologies to connect measurement system nodes. These local nodes may themselves be composed of measurement systems relying on GPIB, USB, and PCI. Physical Ethernet connections are more robust than USB connections, but less so than GPIB or PXI.

LXI (LAN eXtenstions for Instrumentation) is an emerging LAN-based standard. The LXI standard defines a specification for stand-alone instruments with Ethernet connectivity that adds triggering and synchronization features.

Summary: Instrument Bus Performance – Making Sense of Competing Bus Technologies for Instrument Control

Despite the conceptual convenience of designating a single bus or communication standard as the "ultimate" or "ideal" technology, history teaches us that several alternative standards are likely to continue to coexist, since each bus technology has unique strengths and weaknesses. Table 5-1 compiles the performance criteria from the previous section. It should be clear that no single bus is superior across all measures of performance.

	Bandwidth (MB/s)	Latency (µs)	Range (m) (without extenders)	Setup and Installation	Connector Ruggedness
GBIB	1.8 (488.1) 8 (HS488)	30	20	Good	Best
USB	60 (Hi-Speed)	1000 (USB) 125 (Hi-Speed)	5	Best	Good
PCI	132	0.7	Internal PC bus	Better	Better Best (for PXI)
PCI Express	250 (x1) 4000 (x16)	0.7 (x1) 0.7 (x4)	Internal PC bus	Better	Better Best (for PXI)
Ethernet/LAN/ LXI	12.5 (Fast) 125 (Gigabit)	1000 (Fast) 1000 (Gigabit)	100 m	Good	Good

Table 5-1. Bus Performance Comparison

Test system developers can exploit the strengths of several buses and platforms by creating hybrid systems. Hybrid test and measurement systems combine components from modular instrumentation platforms such as PXI and VXI and stand-alone instruments that connect across GPIB, USB, and Ethernet/LAN. One key to creating and maintaining a hybrid system is implementing a system architecture that transparently recognizes multiple bus technologies and takes advantage of an open, multivendor computing platform, such as PXI, to achieve I/O connectivity.

The other key to successfully developing a hybrid system is ensuring that the software you choose at the driver, application, and test system management levels is modular. Though some vendors may offer vertical software solutions for specific instruments, the most useful system architecture is one which breaks up the software functions into interchangeable modular layers so that your system is neither tied to a particular piece of hardware or to a particular vendor. This layered approach provides the best code reuse, modularity, and longevity. For example, VISA (Virtual Instrument Software Architecture) is a vendor-neutral software standard for configuring, programming, and troubleshooting instrumentation

systems comprising GPIB, VXI, serial (RS232/485), Ethernet, USB, and/or IEEE 1394 interfaces. It is a useful tool because the API for programming VISA functions is similar for a variety of communication interfaces.

With hybrid systems, you can combine the strengths of many types of instruments, including legacy equipment and specialized devices. Despite the appeal of finding a one-size-fits-all solution for instrumentation, reality requires that test engineers fit the instruments and associated bus technologies to their specific application needs.

6

Understanding a Modular Instrumentation System for Automated Test

Key Elements for Reducing Cost and Size, Increasing Throughput, and Extending Lifetime

The trends of increasing device complexity and technology convergence are driving test systems to be more flexible. Test systems must accommodate device changes over time, even though cost pressures are demanding longer system lifetimes. The only way to accomplish these objectives is through a software-defined, modular architecture. This white paper will introduce the software-defined concept through virtual instrumentation, provide options for the hardware platform and software implementations, and discuss how a modular system is ideally suited to meet ATE challenges.

Modular Instrumentation – Flexible, User-Defined Software and Scalable Hardware Components

Fundamentally there are two types of instrumentation today, virtual and traditional. Figure 6-1 illustrates the architectures of these types.

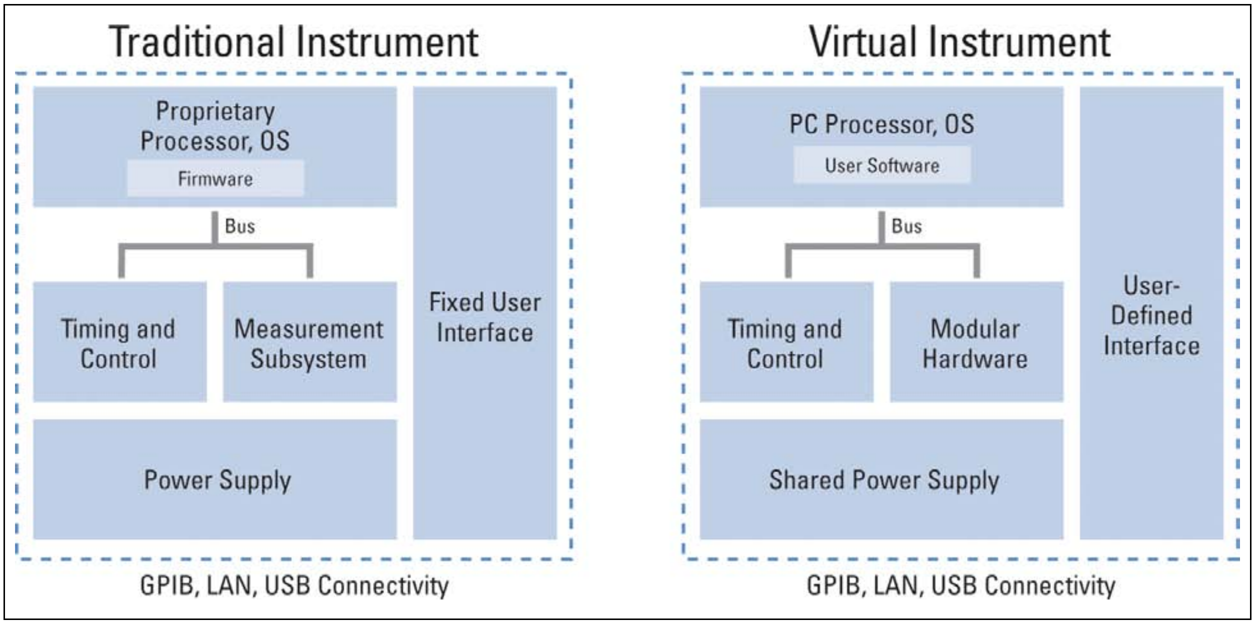

Figure 6-1. Comparing traditional and virtual instrumentation architectures, both share similar hardware components; the primary difference between the architectures is where the software resides and whether it is user-accessible.

The diagrams show the similarities in these two approaches. Both have measurement hardware, a chassis, a power supply, a bus, a processor, an OS, and a user interface. Because the approaches use the same basic components, the most obvious difference from a purely hardware standpoint is how the components are packaged. A traditional, or stand-alone, instrument puts all of the components in the same box for every discrete instrument. An example of a stand-alone instrument is a manual instrument controlled with GPIB, USB, or LAN/Ethernet. These instruments are designed as discrete entities and not designed primarily for system use. While there are a large number of traditional instruments, the software processing and user interface are fixed in the instrument itself and can be updated only when and how the vendor chooses (for example, through a firmware update.) Thus, it is impossible for the user to perform measurements not included in the function list of a traditional instrument, and it makes it challenging to perform measurements for new standards or to modify the system if needs change.

By contrast, a software-defined virtual instrument makes the raw data from the hardware available to users to define their own measurements and user interface. With this software-defined approach, users can make custom measurements, perform measurements for emerging standards, or modify the system if requirements change (for example, to add instruments, channels, or measurements.) While user-defined software can be applied to stand-alone,

application-specific hardware, it is ideally paired with general-purpose, modular hardware where the full flexibility and performance of the measurement software can be exploited. This combination of flexible, user-defined software and scalable hardware components is the core of modular instrumentation.

Modular Hardware for System Scalability

Modular instrumentation can take several forms. In a well designed modular instrumentation system, many of the components – such as the chassis and power supply – are shared across instrument modules instead of duplicating these components for every instrument function. These instrument modules can also include different types of hardware, including oscilloscopes, function generators, digital, and RF. In some cases, as shown in Figure 6-2, the measurement hardware is simply a peripheral that is installed in one of the host computer peripheral ports or peripheral slots. In this case, the host PC provides the processor for performing the measurements in software as well as the chassis for the power supply and I/O.

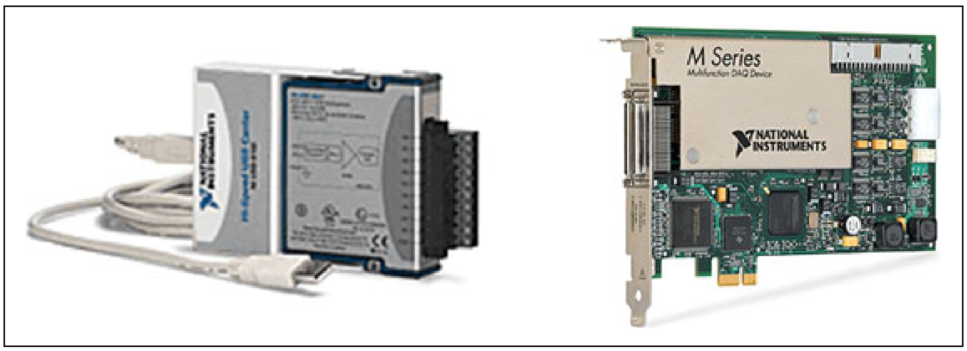

Figure 6-2. Examples of measurement hardware choices for modular instrumentation include a USB peripheral module on the left, and a PCI Express plug-in module on the right.

In other cases, such as PXI (PCI eXtensions for Instrumentation) – a rugged platform for test, measurement and control supported by over 70 member companies – the measurement hardware is housed in an industrial chassis (see Figure 6-3):

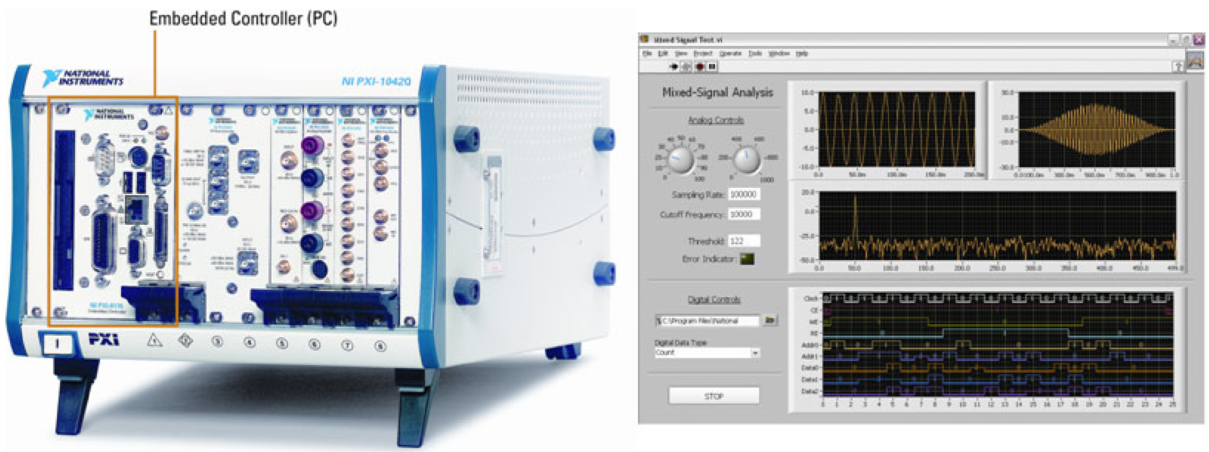

Figure 6-3. This example of a modular instrumentation system uses PXI hardware and NI LabVIEW graphical development software.

In a PXI system, the host computer can be embedded in the chassis (as shown in Figure 6-3) or it can be a separate laptop, desktop, or server that controls the measurement hardware through a cabled interface. Because a PXI system uses the same buses internal to a PC – PCI and PCI Express – and off-the-shelf PC components to control the system, the same modular instrumentation concepts apply equally using a PXI system or a PC. (However, PXI does provide other benefits for modular instrumentation not presented here such as increased channel count, portability, and ruggedness (for more information on PXI, see `ni.com/pxi`.) Regardless of whether the system uses PXI, a desktop PC with internal plug-in modules, or a desktop PC with peripheral I/O modules, this sharing of the chassis and controller greatly reduces cost but also enables the user to control the measurement-and-analysis software. While there are many configuration choices for modular instrumentation, the differentiator between this approach and traditional instrumentation is that the software is open, so that the user can define his/her own measurements as test needs change or as measurements are unavailable on traditional instruments.

It is important to note that this modular approach does not mean that instrument or channel synchronization suffers when compared to traditional instruments that combine functions into a single box. On the contrary, modular instruments are designed to be integrated for system use. All modular instruments provide timing and synchronization capabilities through shared clocks and triggers. For example, for the highest synchronization accuracy, baseband, IF, and RF instruments can synchronize to one another with less than 100 ps interinstrument skew – better than the skew across multiple channels on the same instrument.

Modularity Lowers Cost and Size, Increases Throughput, and Extends Lifetime

While the term "modular" is sometimes misapplied based on the hardware packaging alone, modular instrumentation is about more than just packaging. Users should expect three things of a modular instrumentation system – reduced cost and size through a shared chassis, backplane, and processor; faster throughput through a high-speed connection to the host processor; and greater flexibility and longevity through user-defined software.

As detailed above, all instruments in a modular instrumentation system share the same power supply, chassis, and controller. Stand-alone instruments duplicate the power supply, chassis, and/or controller for every instrument, adding cost and size and decreasing reliability. In fact, every automated test system requires a PC regardless of the bus used; a modular architecture that shares this controller across all the instruments amortizes that cost across the entire system. In modular instrumentation systems, GHz PC processors analyze the data and make measurements using software. The result is measurements at 10 to 100 times the throughput of a test system built solely on traditional instruments, which use built-in vendor-defined firmware and application-specific processors. For example, a typical vector signal analyzer (VSA) performs 0.13 power-in-band measurements/s, whereas an NI modular VSA can perform 4.18 power-in-band measurements/s – a 33X improvement.

Modular instruments require a high-bandwidth, low-latency bus to connect the instrument modules to the shared processor for performing user-defined measurements. While USB provides an excellent user experience in terms of ease of use, PCI and PCI Express (and by extension, the PXI platform, which is based on these buses) provide the highest performance in modular instrumentation. PCI Express today provides slots with up to 4 GB/s, and PXI provides slots with bandwidth up to 2 GB/s each – more than 33 times as fast as HI-Speed USB, 160 times as fast as 100 Mb/s Ethernet, and even 16 times as fast as emerging Gb/s Ethernet (Figure 6-4). Peripheral buses such as LAN and USB always connect to the PC processor through an internal bus such as PCI Express and are therefore, by definition, always lower in performance. As an example of how high-speed buses can impact test and measurement, consider a modular RF acquisition system. A PCI Express x4 (2 GB/s) slot in a desktop PC or PXI system can stream two channels of 100 MS/s, 16-bit IF (intermediate frequency) data directly to a processor for computation. Because neither LAN nor USB can meet these requirements, instruments that need this level of performance always include an embedded, vendor-defined processor to perform the measurements – in which case they are no longer modular.

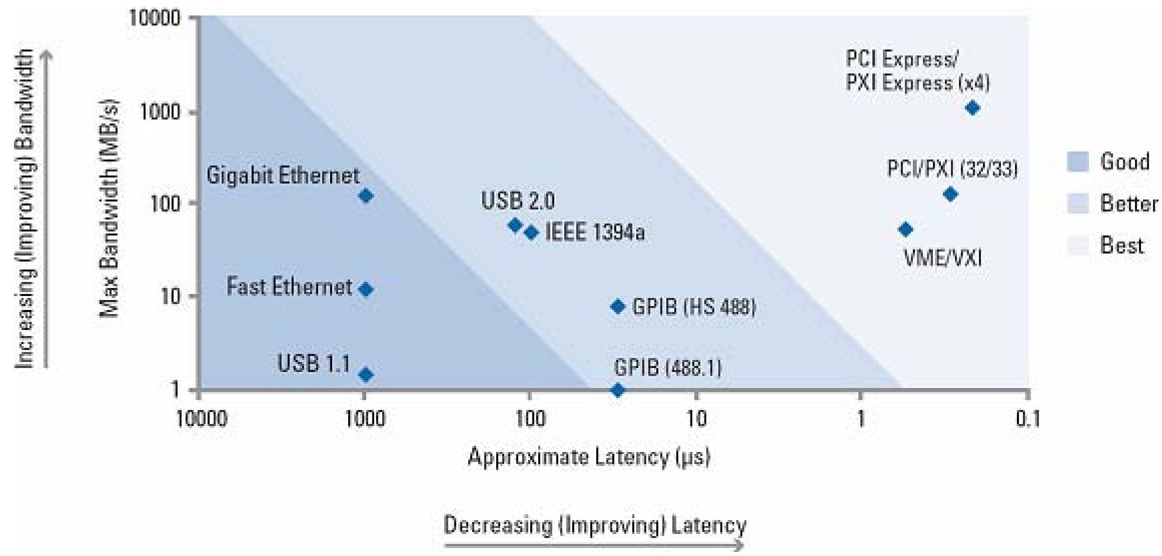

Figure 6-4. PCI and PCI Express provide the highest bandwidth and lowest latency, decreasing test time and delivering flexibility and longevity through user-defined software.

In a modular instrument, the high-speed connection to the host is what delivers flexibility and longevity because it enables the software to reside on the host instead of on the instrument. With the software running on the host, the user and not the vendor defines how the instrument operates. This architecture gives you the power to: 1) make measurements that are not common enough to be included in the typical vendor-defined, nonmodular approach; 2) create measurements for unreleased standards; and 3) define the algorithms used to make specific measurements. The user-defined nature of the software also means that you can add or modify measurements, and even instruments, as the device under test changes. You can also use the direct software access to monitor or control these modular instruments across the network.

It is worth noting that these hardware implementations do not sacrifice measurement performance. Today, the instruments designed with a modular instrumentation approach include industry's highest-resolution digitizer, the highest-bandwidth arbitrary waveform generator, and the most accurate 7½-digit digital multimeter.

Software for Flexible, Custom Measurements

The role of software in modular instrumentation cannot be overstated. Software converts the raw bit stream from hardware into a useful measurement. A well-designed modular instrumentation system considers multiple layers of software, including I/O drivers, application development, and test management, as shown in Figure 6-5.

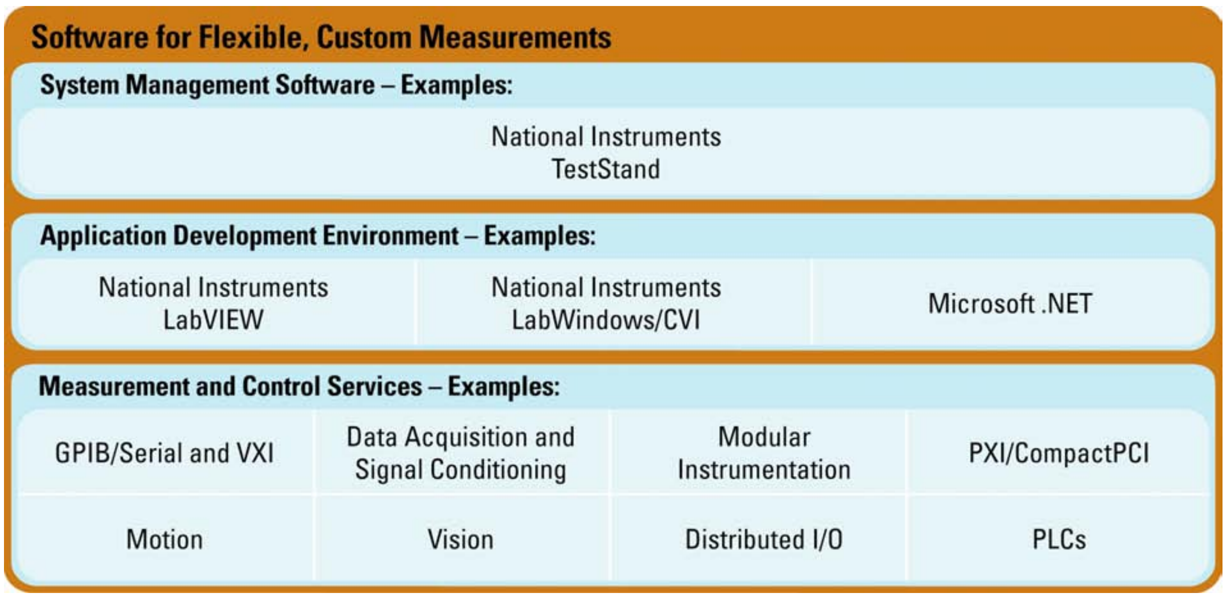

Figure 6-5. Software layers are often used in a modular instrumentation system.

The bottom layer, Measurement and Control Services, is one of the most crucial elements of a modular instrumentation system, though often overlooked. This layer represents the driver I/O software and hardware configuration tools. This driver software is critical, because it provides the connectivity between the test development software and the hardware for measurement and control.

Instrument drivers provide a set of high-level, human-readable functions for interfacing with instruments. Each instrument driver is specifically tailored to a particular model of instrument to provide an interface to its unique capabilities. Of particular importance in an instrument driver is its integration with the development environment so that the instrument commands are a seamless part of the application development. System developers need instrument driver interfaces optimized for their development environment of choice, for example, NI LabVIEW, C, C++, or Microsoft .NET.

Also represented in Measurement and Control Services are configuration tools. These configuration tools include resources for configuring and testing I/O, as well as storing scaling, calibration, and channel-aliasing information. These tools are important for quickly building, troubleshooting, and maintaining an instrumentation system.

The software in the Application Development Environment layer provides the tools to develop the code or procedure for the application. Although graphical programming is not a requirement of a modular instrument system, these systems often uses graphical tools for their ease of use and rapid development. Graphical programming uses "icons" or symbolic functions that pictorially represent the action to be performed, as shown in Figure 6-6. These symbols are connected together through "wires" that pass data and determine order of execution. LabVIEW provides the industry's most used and most complete graphical development environment.

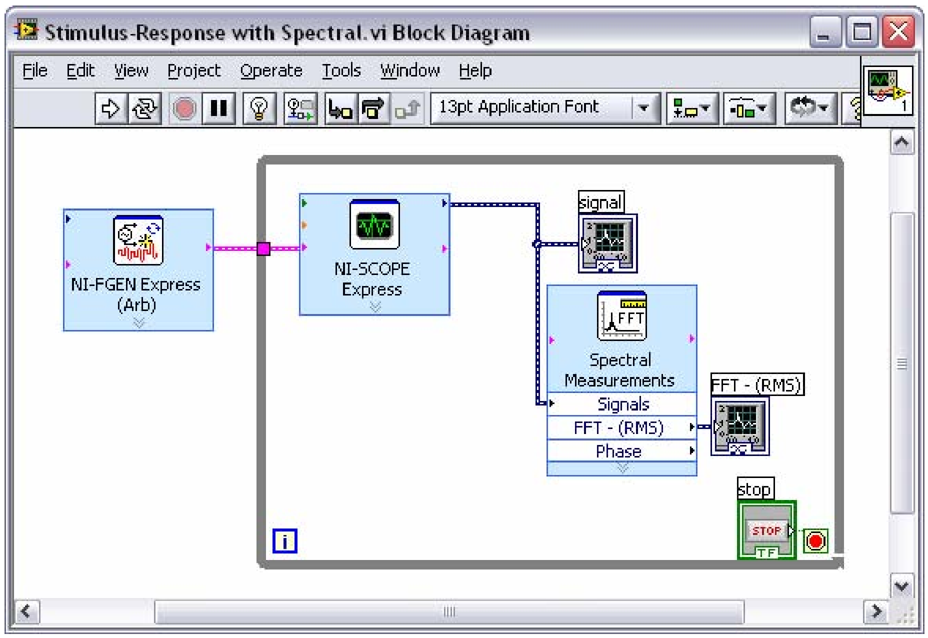

Figure 6-6. Code for a typical stimulus/response application using modular instrumentation, written in LabVIEW, 1) generates a signal from an arbitrary waveform generator; 2) acquires the signal with a digitizer/oscilloscope; 3) performs a fast Fourier transform; and 4) graphs the result of the FFT on the user interface (front panel).

Some applications also require an additional layer of software management, either for test execution or visibility into test data. These are represented in the System Management Software layer. For highly automated test systems, test management software provides a framework for sequencing, branching/looping, report generation, and database integration. The test management tool must also provide tight integration into the development environments where the application-specific code is created. NI TestStand, for example, provides this framework for sequencing, branching, report generation, and database integration and includes connectivity to all common development environments. In other applications that need visibility into large quantities of test data, other tools may be useful. These needs include quick access to large volumes of scattered data, consistent reporting, and data visualization. These software tools aid in managing, analyzing, and reporting data collected during data acquisition and/or generated during simulations. Every layer of this software architecture should be considered for a modular instrumentation system.

Modular Instrumentation – Meeting the Needs of Automated Test

As devices become more complex and include more disparate technologies, test systems must become more flexible. While test systems must accommodate devices changing over time, cost pressures demand longer system lifetimes. The only way to accomplish these objectives is through a software-defined, modular architecture. Through shared components, high-speed buses, and open, user-defined software, modular instrumentation is best suited to meet the needs of ATE today and the future.

7

PXI – The Industry Standard Platform for Instrumentation

PXI (PCI eXtensions for Instrumentation) is a rugged PC-based platform for measurement and automation systems. PXI combines PCI electrical-bus features with the rugged, modular, Eurocard packaging of CompactPCI, and then adds specialized synchronization buses and key software features. PXI is both a high-performance and low-cost deployment platform for measurement and automation systems. These systems serve applications such as manufacturing test, military and aerospace, machine monitoring, automotive, and industrial test.

Developed in 1997 and launched in 1998, PXI was introduced as an open industry standard to meet the increasing demands of complex instrumentation systems. Today, PXI is governed by the PXI Systems Alliance (PXISA), a group of more than 70 companies chartered to promote the PXI standard, ensure interoperability, and maintain the PXI specification. For more information on the PXISA, including the PXI specification, visit `www.pxisa.org`.

Hardware Architecture

PXI systems are comprised of three basic components – chassis, system controller, and peripheral modules.

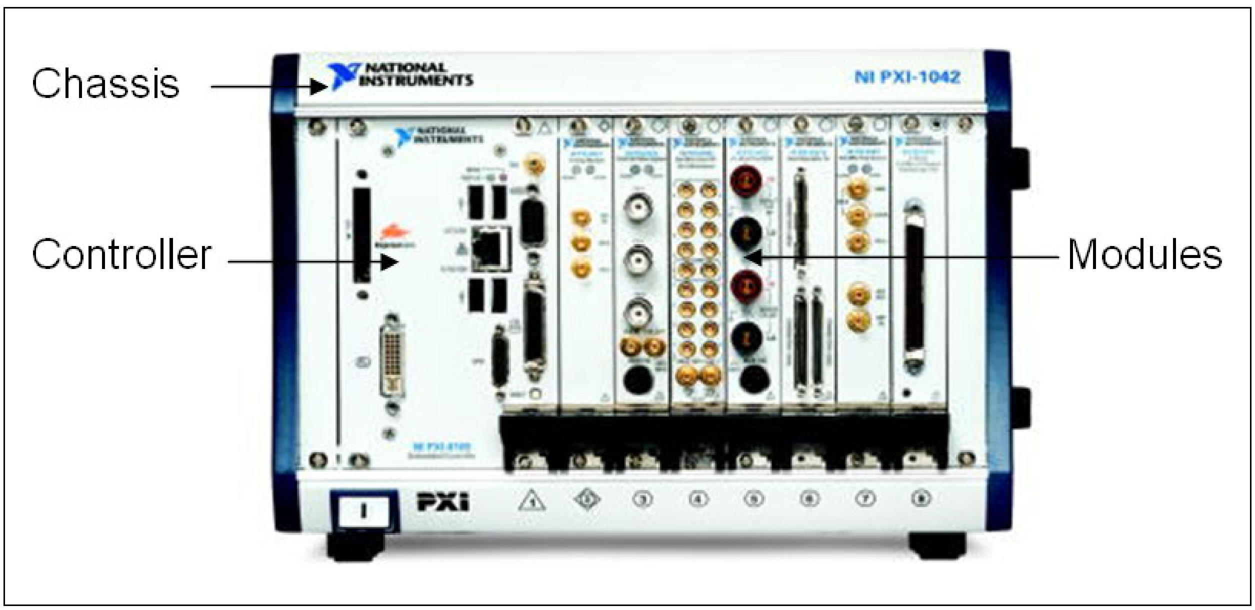

Figure 7-1. A standard 8-slot PXI chassis contains an embedded system controller and seven peripheral modules.

PXI Chassis

PXI chassis provides the rugged, modular packaging for the system. Chassis available in both 3U and 6U sizes, generally ranging in size from 4-slots to 18-slots, are available with special features such as DC power supplies and integrated signal conditioning. The chassis contains the high-performance PXI backplane, which includes the PCI bus and timing and triggering buses. Using these timing and triggering buses, users can develop systems for applications requiring precise synchronization.

PXI Controllers

As defined by the PXI Hardware Specification, all PXI chassis contain a system controller slot located in the leftmost slot of the chassis (slot 1). Controller options include remote controllers from a desktop, workstation, server, or a laptop computer and high-performance embedded controllers with either a Windows 2000/XP or a real-time OS (LabVIEW Real-Time)

PXI Remote Controllers

There are two types of PXI remote controllers:

- Laptop control of PXI
- PC control of PXI

Laptop Control of PXI: With ExpressCard MXI (Measurement eXtensions for Instrumentation) and PCMCIA CardBus interface kits, users can control PXI systems directly from laptop computers. During boot-up, the laptop computer will recognize all peripheral modules in the PXI system as PCI devices. Using ExpressCard MXI you can control your PXI system with a sustained throughput of up to 214 MB/s.

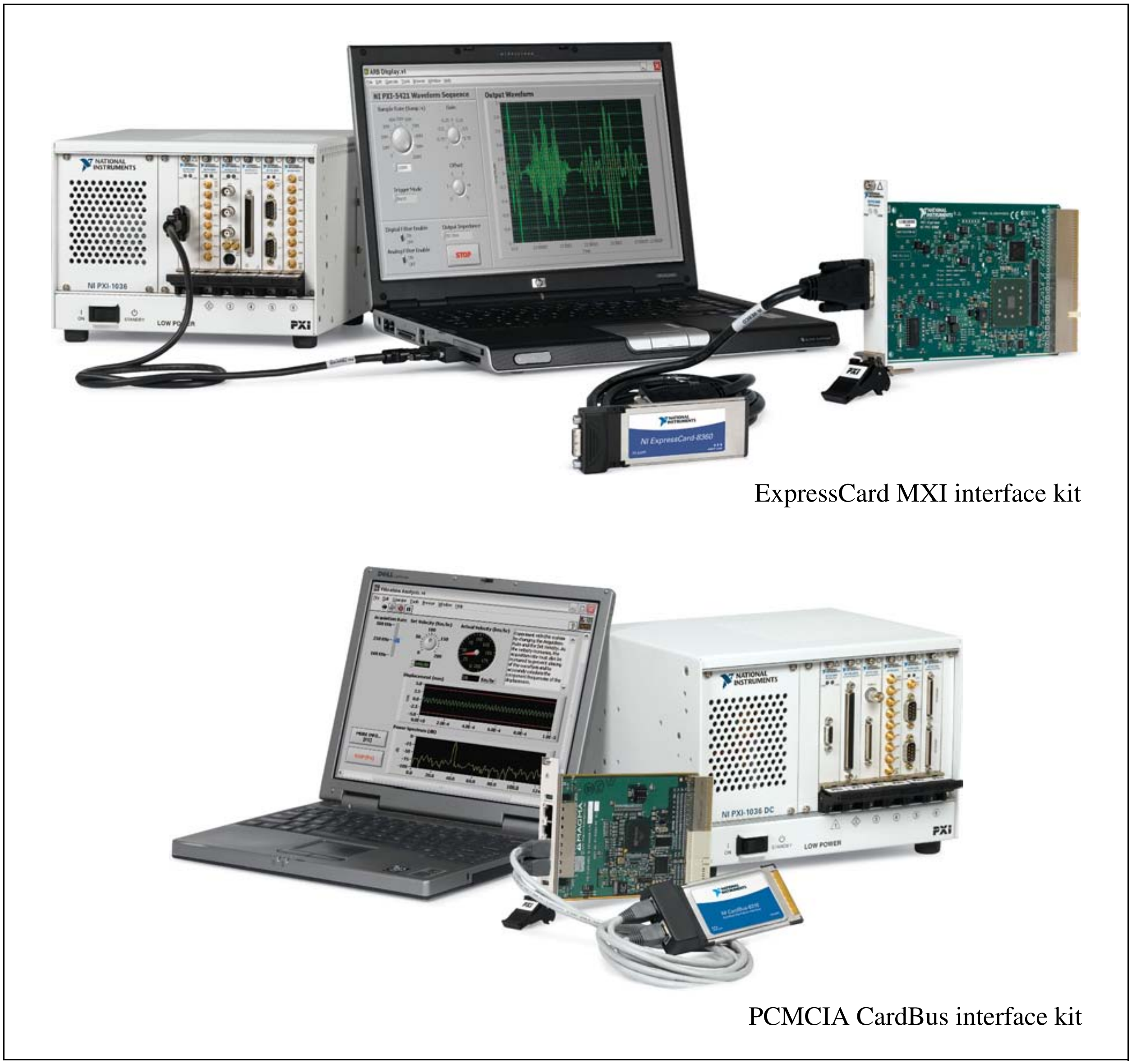

Figure 7-2. Laptop Control of PXI

Users now have the advantage of mobile PXI systems for applications such as field tests, in-vehicle data logging, NVH and NDT with laptop control of PXI. You can purchase any ExpressCard MXI compatible laptop or PCMCIA CardBus compatible laptop to remotely control your PXI system. For more information please refer to Laptop control of PXI.

PC Control of PXI: With MXI-Express and MXI-4 interface kits, users can control PXI systems directly from desktop, workstation, or server computers. During boot-up, the computer will recognize all peripheral modules in the PXI system as PCI devices.

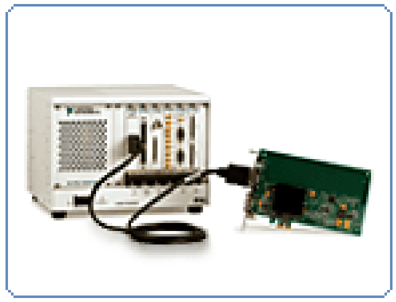

Figure 7-3. Remote control with 2-port MXI-Express provides simultaneous control of two PXI chassis with combined throughput of 160 MB/s.

Using MXI-Express you can control your PXI system with a sustained throughput of up to 832 MB/s. With the MXI-Express 2-port interface kit, users can control two PXI systems simultaneously from a single PC.

Figure 7-4. Remote control with MXI-4 provides PC control of PXI, as well as multichassis PXI systems.

MXI-4 interface kit comes with low-cost copper links or fiber-optic links for both extended distances and electrical isolation. As shown in Figure 7-4, you can build multichassis PXI systems with MXI-4 for high-channel-count applications. Using a MXI-4 link, you can implement either a daisy-chain or a star topology to build multichassis systems. For more information on topologies for multichassis configurations, refer to the MXI-4 Series User Manual. You can purchase any desktop, workstation or server computer, and then remotely control your PXI system using either MXI-Express or copper/fiber optic MXI-4 serial link. For more information please refer to PC control of PXI.

With PXI remote controllers, you can maximize processor performance with minimized cost by using a desktop computer or laptop to remotely control a PXI system. Because all remote control products are software transparent, no additional programming is required.

PXI Embedded Controllers

Embedded controllers eliminate the need for an external PC, therefore providing a complete system contained in the PXI chassis. PXI embedded controllers are typically built using standard PC components in a small, PXI package. For example, the NI PXI-8105 controller has the 2.0 GHz Intel Core Duo T2500 dual-core processor, with up to 2 GB of DDR2 RAM, a hard drive, and standard PC peripherals such as ExpressCard, Hi-Speed USB, Ethernet, serial, parallel, and GPIB ports. There are two types of PXI embedded controllers

- PXI Embedded Controllers with Windows
- PXI Embedded Real Time Controllers

PXI Embedded Controllers with Windows: PXI embedded controllers with windows come with standard PC features such as integrated CPU, hard-drive, RAM, Ethernet, video, keyboard/mouse, serial, USB, and other peripherals, as well as Microsoft Windows and all device drivers already installed. Because the controllers have Microsoft Windows, the user experience is no different than a PC or laptop computer. It has similar application software available in your PC or laptop computer such as Microsoft Office Word, Excel, and PowerPoint.

PXI Embedded Real Time Controllers: PXI Embedded Real Time Controllers also come with standard PC features along with a Real-Time OS such as LabVIEW Real-Time or VxWorks to deliver real-time, deterministic, and reliable I/O for measurement, automation, and control. Because RT Series PXI controllers are configured and programmed over the Ethernet, you can distribute a real-time application across the network and remotely monitor it. These controllers are designed for applications requiring deterministic, reliable performance and can be run headless (i.e. no keyboard, mouse or monitor).

Figure 7-5. National Instruments PXI-8105 2.0 GHz Dual-Core PXI Embedded Controller. Notice the familiar PC peripherals such as keyboard/mouse and monitor connections, as well as the hard drive, USB, Ethernet, serial, ExpressCard and other standard PC peripherals. This controller runs standard Windows 2000/XP OSs, or with LabVIEW Real-Time.

Embedded controllers are ideal for portable systems and contained "single-box" applications where the chassis is moved from one location to the next.

PXI Peripheral Modules

National Instruments offers more than 100 different PXI modules; and because PXI is an open industry standard, nearly 1200 products are available from the 70+ members of the PXI Systems Alliance.

- Analog Input and Output
- Boundary Scan
- Bus Interface and Communication
- Carrier Products
- Digital Input and Output
- Digital Signal Processing
- Functional Test and Diagnostics

- Image Acquisition
- Prototyping Boards
- Instruments
- Motion Control
- Power Supplies
- Receiver Interconnect Devices
- Switching
- Timing Input and Output
- RF and Communications

Because PXI is directly compatible with CompactPCI, you can use any 3U CompactPCI module in a PXI system. A categorized list of modules offered by National Instruments and our PXI product partners is available at `ni.com/pxi`.

PXI also preserves investments in stand-alone instruments or VXI systems by providing standard hardware and software for communication to these systems. For example, interfacing a PXI system to GPIB-based instrumentation is no different with a PXI-GPIB module than it is with a PCI-GPIB module. The software is identical. Additionally, a number of methods are available to build hybrid systems interfacing PXI, USB, LAN/LXI, VXI and stand-alone instruments.

Software Architecture

Because PXI hardware is based on standard PC technologies, such as the PCI bus, as well as standard CPUs and peripherals, the standard Windows software architecture is familiar to users as well. Development and operation of Windows-based PXI systems is no different from that of a standard Windows-based PC. Additionally, because the PXI backplane uses the industry-standard PCI/PCI Express bus, writing software to communicate with PXI devices is, in most cases, identical to that of PCI devices. For example, software to communicate to an NI PXI-6251 multifunction data acquisition module is identical to that of an PCI-6251 board in a PC. Therefore, existing application software, example code, and programming techniques do not have to be rewritten when moving software between PC-based and PXI-based systems.

Figure 7-6. Two different packages – one software standard. In software, communication with a PXI module (bottom) is identical to that with a PCI board (top).

As an alternative to Windows-based systems, you can use a real-time software architecture for time-critical applications requiring deterministic loop rates and headless operation (no keyboard, mouse, or monitor). Additional information on using LabVIEW Real-Time with PXI systems is available at `www.ni.com/realtime`.

PXI – Industry Standard for Modular Instrumentation

Every bus is unique and has its advantages. For example, USB is excellent for easy desktop connectivity; LAN/Ethernet is well-suited for distributed systems; and PCI and PCI Express provide high performance for ATE. For applications demanding a modular solution, users should expect reduced cost and size through a shared chassis, backplane, and processor; faster throughput through a high-speed connection to the host processor; and greater flexibility and longevity through user-defined software.

PXI, based on PCI and next-generation PCI Express, is the fastest-growing test and measurement standard since GPIB. PXI best meets modular instrumentation demands now and in the future, with more than 70 vendors in the PXI Systems Alliance, more than 1,200 products available today, and a projected 25 percent annual growth rate through 2011 (Frost & Sullivan, 2005). Primarily, all instruments in a PXI system share the same power supply, chassis, and controller. Alternative approaches duplicate power supply, chassis, and/or controller for every instrument, adding cost and size and decreasing reliability. With PXI, the controller can be a high-performance slot 0 embedded controller, desktop PC, laptop, or server-class machine. When you require faster processing, you can easily upgrade the controller of a PXI system. To reuse existing equipment, you can use PXI to control USB, GPIB, LAN/LXI, serial, and VXI instruments.

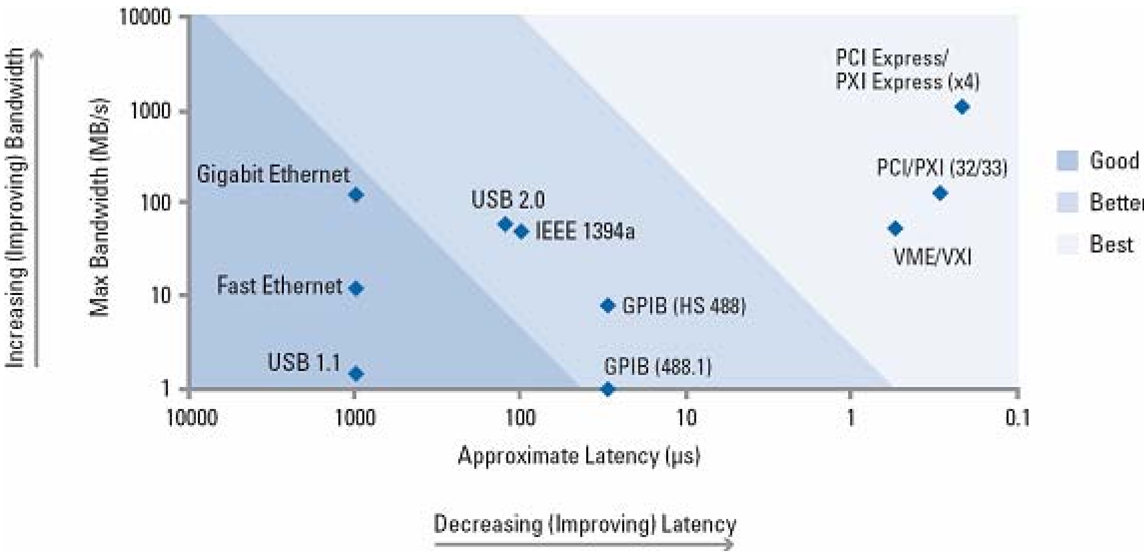

Figure 7-7. PCI and PCI Express provide the highest bandwidth and lowest latency, decreasing test time and delivering flexibility and longevity through user-defined software.

Modular instruments require a high-bandwidth, low-latency bus to connect instrument modules to the shared processor for performing user-defined measurements. PXI meets these needs with bandwidth up to 2 GB/s for each slot. Take a modular RF acquisition system for example. PXI can stream two channels of 100 MS/s, 16-bit IF data directly to a processor for computation. Neither LAN nor USB can meet these requirements, so these instruments always include an embedded, vendor-defined processor. Hence high-bandwidth standards such as PXI provide a truly software-defined approach required for modular instrumentation.

Why Customers Choose PXI?

Higher Throughput

Every application is unique and has very specific needs. However, bandwidth and latency are two important attributes of a platform for many applications. Latency tends to dominate single-point operations, such as digital multimeter/switch scanning, and bandwidth tends to dominate data streaming applications, such as waveform stimulus/response. PXI provides high speed for a wide range of applications with both high bandwidth and low latency via the PCI/PCI Express bus as shown in figure 7-7.

Timing and Synchronization

Many measurement and automation applications require advanced timing and synchronization capabilities that you cannot implement directly across PC standard I/O buses like PCI/PCI Express, Ethernet/LAN, USB, and so on. PXI offers advanced timing and synchronization features to meet your application needs:

- 100 MHz differential system reference clock
- 10 MHz reference clock signal
- Differential star trigger
- Star trigger bus with matched-length trigger traces to minimize intermodule delay and skew
- Trigger bus to send and receive high-speed timing and triggering signals
- Differential signals for multichassis synchronization

System Reliability

The PXI specification defines requirements that make PXI systems well-suited for harsh environments. PXI features the high-performance IEC (International Electrotechnical Commission) connectors and rugged Eurocard packaging system used by CompactPCI. The PXI specification also defines specific cooling and environmental requirements to ensure operation in industrial environments. Modularity makes it easy to configure, reconfigure, and repair your PXI systems, resulting in very low mean time to repair (MTTR). Because PXI is modular, you can update individual modules and components without replacing the entire system.

Lower System Costs

Because PXI is a PC based platform, it delivers the high-precision instrumentation, synchronization, and timing features at an affordable price. The low cost of PC components is only the beginning of the savings you gain from using PXI. With PXI, you use the same OS and application software such as MS Excel and Word in your office and on the production floor. The familiarity of the software eliminates training costs and the need to retrain personnel every time you implement a new system. Because the foundation of PXI is PC technology, you benefit from low component costs, familiar software, and system reuse.

Expansion of the PXI Platform – PXI Express

PXI Express technology is the latest addition to the PXI platform. The PXI Express Specification integrates PCI Express signaling into the PXI standard, which increases backplane bandwidth from 132 MB/s to 6 GB/s, a 45X improvement. It also enhances PXI timing and synchronization features by incorporating a 100 MHz differential reference clock and differential triggers.

The PXI Express specification adds these features to PXI while maintaining backward compatibility:

Software: PCI Express uses the same operating system and driver model as PCI, resulting in complete software compatibility between PCI-based systems (such as PXI) and PCI Express-based systems (such as PXI Express). This software compatibility is ensured by the PCI Special Interest Group (PCI-SIG), a group composed of member companies, such as Intel, who are committed to the development and enhancement of the PCI and PCI Express standards.

Hardware: PXI Express chassis provide hybrid peripheral slots that accept both PXI Express peripheral modules and hybrid slot-compatible PXI peripheral modules. These peripheral slots deliver signaling for both PCI and PCI Express.

You can use code you have written for previous PXI systems with PXI Express systems because PXI Express maintains complete software compatibility with PXI. Software compatibility includes OSs such as Windows XP and Linux, application software such as Microsoft Office, and user code such as LabVIEW VIs and C++ projects. For more information please refer PXI Express.

Conclusion: PXI – The Industry Standard Platform for Instrumentation

PXI modular instrumentation defines a rugged computing platform for measurement and automation users that clearly takes advantage of the technology advancements of the mainstream PC industry. By using the standard PCI/PCI Express bus, PXI modular instrumentation systems can benefit from widely available software and hardware components. The software applications and OSs that run on PXI systems are already familiar to users because they are already used on common desktop computers. PXI meets your needs by adding rugged industrial packaging, plentiful slots for I/O, and features that provide advanced timing and triggering capabilities.

Strategies for Improving Test System Performance

8

Maximizing Throughput in an Automated Test System

"How do I maximize my automated test system throughput?" is a common question posed by many engineers and scientists. For years, engineers have employed numerous strategies to extract more speed from their systems in both R&D laboratories and on the manufacturing floor. These optimization techniques have often included brute force procedures, such as cutting down the number of tests and purchasing redundant instruments. This paper describes four strategies for maximizing the throughput without having to make such sacrifices. These strategies include:

- Strategy 1: Choose Highest-Throughput Bus for Your Application
- Strategy 2: Select Software that Takes Full Advantage of the Latest Processor Technology
- Strategy 3: Use Hardware Synchronization
- Strategy 4: Design a System Architecture that Supports Parallel Test and Resource Sharing

Strategy 1: Choose the Highest-Throughput Bus for Your Application

On the surface, it may appear straightforward to select a bus based on its bandwidth alone. The theoretical bandwidth might give an indication of performance, but, unfortunately, it is not that simple. In reality, there are four major factors that affect instrument bus performance – bandwidth, latency, implementation, and application. And because most test-industry buses are based on PC buses, they follow the PC trend that system buses such as PCI and PCI Express perform better than communication buses such as USB and LAN.

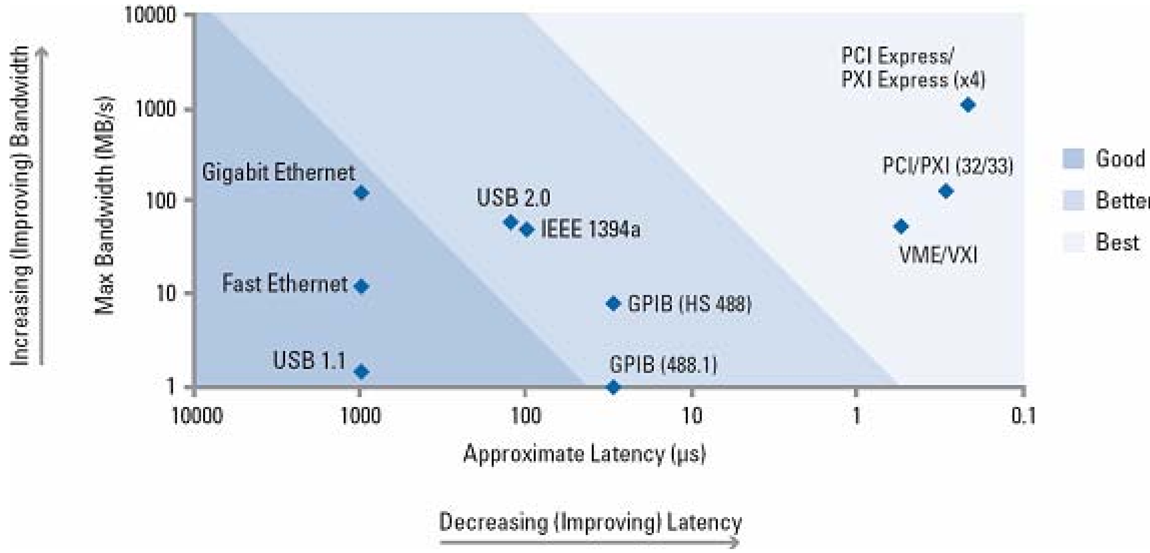

Figure 8-1. Bandwidth versus Latency of Mainstream Test and Measurement Buses

A lot of bus performance comparisons focus solely on the bus bandwidth and ignore the other hardware and software components that influence actual bus performance. The bandwidth is the data rate. This is usually measured in millions of bytes per second (MB/s). The latency is the transfer time. This is usually measured in microseconds (μs). For example, in Ethernet transfers, large blocks of data are broken into small segments and sent in multiple packets. The latency is the amount of time to transfer one of these packets. Figure 8-1 compares the theoretical bandwidths versus latencies of mainstream test and measurement buses.
The implementation of the bus software, firmware, and hardware affects its performance. Not all instruments are created equal. A PC implemented with a faster processor and more RAM performs better than one with a slower processor and less RAM. The same holds true for instruments. The implementation trade-offs made by the instrument designer, whether working with a user-defined virtual instrument or vendor-defined traditional one, have an impact on the instrument performance. One of the main benefits of virtual instruments is that the end user, as the instrument designer, decides the optimal implementation trade-offs.

The final major factor affecting bus performance is the application or how the instrument is used. The instrument I/O hardware and firmware, CPU/RAM combination, software application, and measurement speed affect the bus performance. Changing any one of these components may change the bus performance, just like changing from one bus to another affects overall system performance. In some applications, the measurement subsystem is the bottleneck, and in others, the bottleneck is the processor subsystem. Understanding which subsystem is the bottleneck provides a path to improving performance. And the key point remains that in order for a particular bus to be high-performance or even usable, it must exceed the application requirements.

These factors impact whether an instrument bus exceeds required performance. Use benchmarks to compare actual performance between potential instruments.

Strategy 2: Choose Software that Takes Full Advantage of the Latest Processor Technology

Multicore processors are the latest innovation in the PC industry. These first multicore processors contain two cores, or computing engines, located in one physical processor – hence the name dual-core processors. Processors with more than two cores also are on the horizon. Dual-core processors can simultaneously execute two computing tasks. This is advantageous in multitasking environments, such as Windows XP, in which you simultaneously run multiple applications. Two applications – National Instruments LabVIEW and Microsoft Excel, for example – each can access a separate processor core at the same time, thus improving overall performance for applications such as data logging.

The other major performance advantage of dual-core processors is gained by applications with multithreading. Multithreading gives an application the ability to separate its tasks into individual threads. A dual-core processor can simultaneously execute two of these threads. Dual-core PCs demonstrate significant performance improvements, especially for multithreaded applications. Benchmarks in NI LabVIEW 8 demonstrate a performance improvement for single-threaded applications of up to 25 percent between the National Instruments PXI-8105 dual-core embedded controller and the NI PXI-8196 single-core embedded controller (2.0 GHz Intel Pentium M processor 760), which have equivalent processor clock rates. This improvement is a result of numerous enhancements in the processor and chipset between these two generations of Intel architectures. You can see the performance improvement resulting from the fact that the PXI-8105 processor is dual-core in the multithreaded application benchmarks, which demonstrate an improvement of up to 100 percent compared to the PXI-8196 embedded controller.

Multithreading can significantly improve the performance of a test system. During the testing of a unit under test (UUT), some actions are independent in terms of their functionality and resource usage. For example, when testing a device, you may need to initialize several instruments. The initialization of different instruments is totally independent. Thus, you can break those tasks into multiple threads to improve performance. To fully take advantage of this capability, you must choose software that supports multithreading from driver level and application level to test executive level. For example, you need to choose hardware devices that include multithreaded software drivers. For this reason, all NI hardware devices are released with multithreaded driver software.

Also, you should consider the skill level required to take advantage of these multicore processors. For example, text-based programming languages include built-in multithreading libraries to create and manage threads. However, in text-based languages, where code typically runs sequentially, it often can be difficult to visualize how various sections of code run in parallel. Because the language syntax is sequential, the code basically runs line by line. And because threads within a single program usually share data, communication among threads to coordinate data and resources is so critical that you must implement it carefully to avoid incorrect behavior when running in parallel. You must write extra code to

manage these threads. Thus, the process of converting a single-threaded application into a multithreaded one can be time-consuming and error-prone.

Text-based programming languages must incorporate special synchronization functions when sharing resources such as memory. If you do not implement multithreading properly in the application, you may experience unexpected behavior. Conflicts can occur when multiple threads request shared resources simultaneously or share data space in memory. Current tools, such as multithread-aware debuggers, help a great deal, but in most cases, you must carefully keep track of the source code to prevent conflicts. In addition, you must often adapt your code to accommodate parallel programming.

In contrast, graphical programming tools such as LabVIEW can easily represent parallel processes and are inherently multithreaded. For example, two independent loops running without any dependencies automatically execute in separate threads.

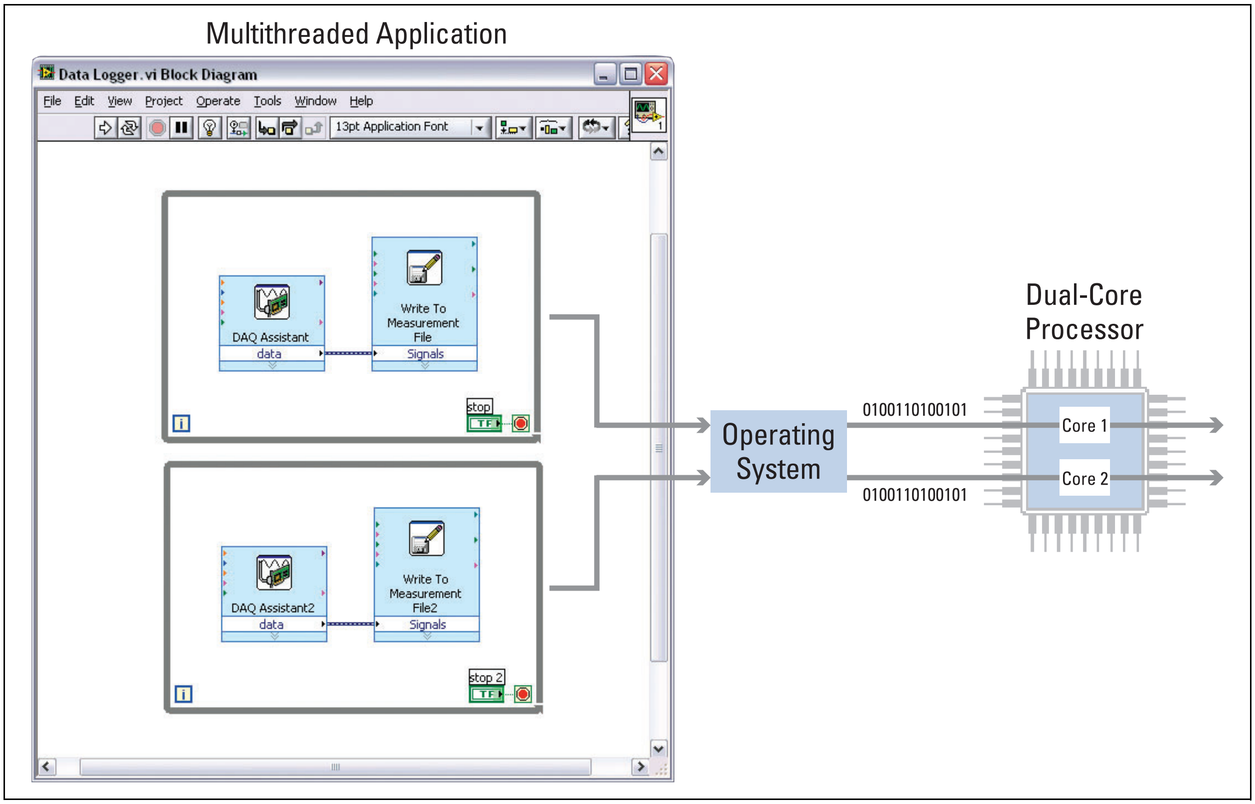

Figure 8-2. The parallel nature of graphical programming automatically implements multithreading on separate processing cores.

On computers with dual-core processors, multithreaded LabVIEW VIs can handle performance increases without any intervention on your part. The multiple threads are scheduled to run on separate processor cores without external instruction.

Strategy 3: Use Hardware Synchronization

Nearly all automated test systems require engineers to synchronize two or more instruments or switches. For example, in a fuel cell test application, an engineer must program a digital multimeter to scan through hundreds of individual cells using a switch. There are two types of synchronization to choose from: software-timed and hardware-timed. Software-timed synchronization uses triggers generated by a software call, and hardware-timed synchronization uses triggers generated by signals connected to the hardware trigger lines. With software synchronization, there is significant software dependency and overhead because software-timed systems need software intervention to advance the trigger for each reading. The timing of these scans is subject to the performance of the software system because the software shares resources with all other applications that require processor time.

Test throughput is maximized by implementing hardware-timed synchronization, which removes the processor dependency. Hardware synchronization takes advantage of triggers from both instruments trying to communicate. Using the DMM/switch example above, instead of adding software delay to allow for the switch settling time, the instrument and switch interact by sending triggers to each other when complete or ready. The instrument sends a trigger signal when the reading is complete, and the switch sends a trigger once it has settled and is ready for the next measurement. All trigger interaction is hardware-controlled, which minimizes time wasted between measurements and guarantees maximum throughput. Hardware synchronization between the switch and the instrument is completely independent of the software environment and is not affected by software.

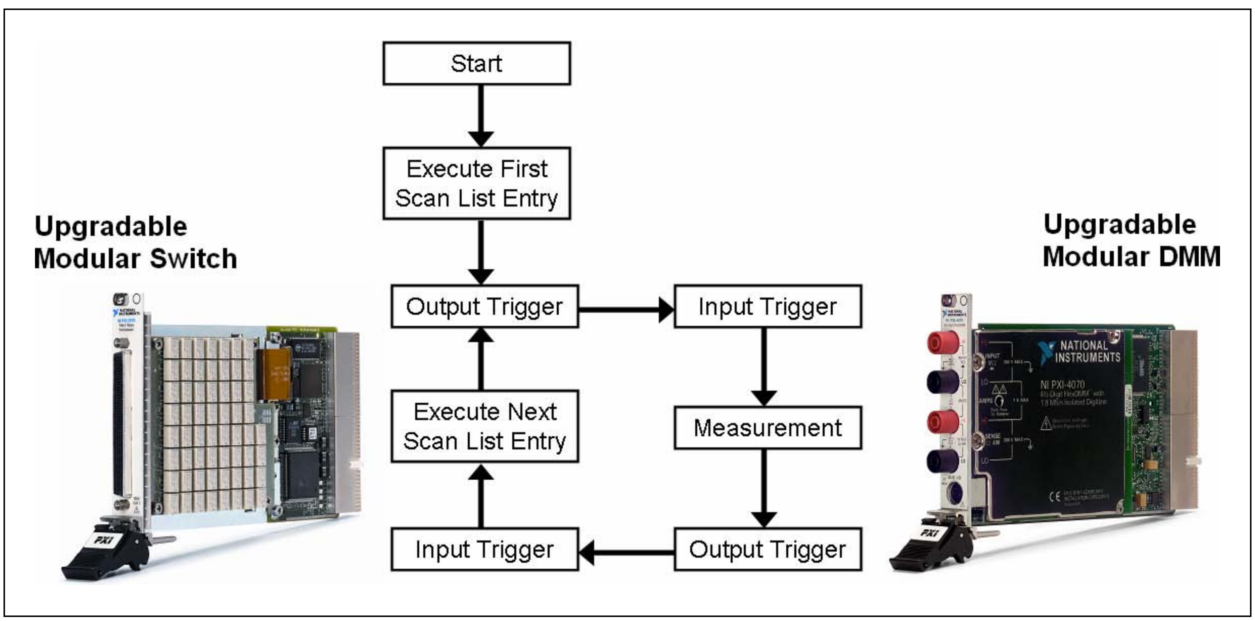

Figure 8-3. Hardware Synchronization Scheme for a Multimeter and Switch

An example of hardware synchronization is shown in Figure 8-3. In this example, the NI PXI-4070 FlexDMM communicates with the NI PXI-2530 high-channel multiplexer switch. After the first channel is closed, the switch trigger tells the DMM to take a measurement. When the DMM completes the measurement, its trigger tells the switch to move on to the next channel, and the cycle repeats without software intervention. Because of this determinism, additional software activity (such as adding extra data processing) does not affect the results of the hardware handshaking test as it does when software scanning is used.

The execution time improvement of hardware handshaking can be seen in Table 8-1. These are benchmark results for 1,000 switched DMM scans using an NI PXI-2501 FET switch and a PXI-4070 FlexDMM. The use of hardware handshaking increased throughput more than 46 percent. Also note that the hardware-handshaking use case is more resilient to extra software processing tasks, which severely slows execution of software scanning.

	Standard Run	**With Extra Processing**	**% Change**
Software Scanning	3021 ms	3840 ms	27
Hardware Handshaking	1640 ms	1655 ms	0.9
% Throughput Increase Using Hardware Handshaking	**46**	**57**	—

Table 8-1. Execution Time Comparison for 1,000 Switched DMM Scans

The determinism and throughput of the system is maximized with hardware synchronization. Additionally, if PXI instrumentation is used, then all trigger signals are passed over the PXI synchronization backplane, eliminating the need for any external wiring and simplifying setup.

Strategy 4: Design a System Architecture that Supports Parallel Test and Resource Sharing

You may have explored ways to enhance test system throughput in the past though parallel testing. However, the latest off-the-shelf test management software tools simplify parallel test system implementation. These tools increase test throughput and drive down test system costs. In general, parallel testing involves testing multiple products or subcomponents simultaneously. A parallel test station typically shares a set of test equipment across multiple test sockets, but, in some cases, it may have a separate set of hardware for each UUT. The majority of nonparallel test systems test only one product or subcomponent at a time, leaving expensive test hardware idle more than 50 percent of the test time. Thus, with parallel testing, you can increase the throughput of manufacturing test systems without spending a lot of money to duplicate and fan out additional test systems. The following sections discuss ways parallel testing can reduce the cost of test and describe various approaches for implementing parallel testing in your test systems.

Choosing a Parallel Test Architecture

While you can implement parallel testing in most existing test systems, modular test system architectures deliver the best results when used in a parallel testing environment. Test management software, such as NI TestStand, and modular PXI hardware components offer many features for obtaining the highest performance out of a parallel test system. However, you can implement parallel testing using much of your existing test hardware without further hardware investment. Once you have selected your test architecture, the next step is to select the best process model based on your desired UUT test behavior.

Common Parallel Process Modules

When testing the proper assembly or functionality of a UUT, there are a variety of tasks to perform in any test system. These tasks include a mix of model or family-specific tests as well as many procedures that have nothing to do with actually testing the UUT. A process model separates the system-level tasks from the UUT-specific tests to significantly reduce development efforts and increase code reuse. Some of the tasks that a process model handles are tracking the UUT identification number, initializing instruments, launching test executions, collecting test results, creating test reports, and logging test results to a database. NI TestStand provides two process models, the parallel process model and the batch process model, to facilitate the general test flow of parallel testing based on your UUT test requirements.

You can use a parallel process model to test multiple independent test sockets. With this model, you can start and stop testing on any UUT at any time. For example, you might have five test sockets for performing radio board tests. Using the parallel process model, you can load a new board into an open socket while the other sockets test other boards. Figure 8-4 illustrates how the parallel process executes.

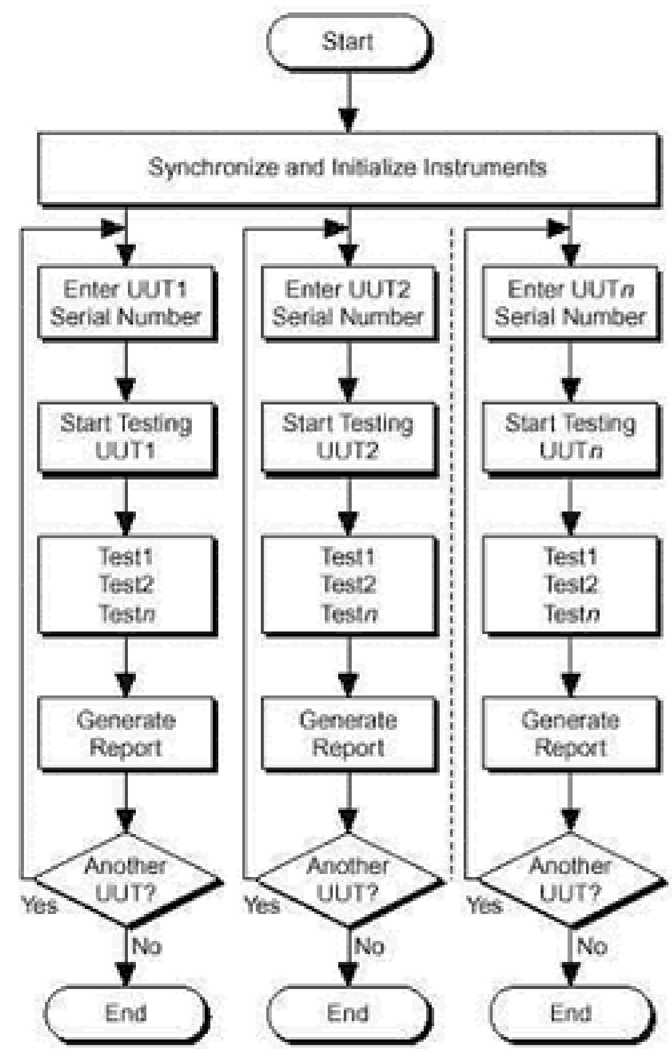

Figure 8-4. Parallel Process Model Flow Chart

Alternatively, you can use a batch process model to control a set of test sockets that test multiple UUTs as a group. For example, you might have a set of circuit boards attached to a common carrier. The batch model ensures you can start and finish testing all boards at the same time. The batch model also provides batch synchronization features. For instance, you can specify that the step runs only once per batch if a particular step applies to the batch as a whole. With a batch process model, you can also specify that certain steps or groups of steps cannot run on more than one UUT at a time or that certain steps must run on all UUTs simultaneously.

Instrument Sharing

If you are trying to increase your test system performance while lowering your cost, providing each test socket with a dedicated set of instruments is not a feasible solution. Implementing a parallel test system often does not require any additional hardware investment. With parallel testing, you can share existing instrumentation in the test system among multiple test sockets. Decreasing idle time during a UUT test cycle provides substantial performance improvements without additional hardware costs. In many cases, you can add other inexpensive instruments to further optimize overall system performance while sharing the more expensive hardware among the test sockets.

Prior to the availability of off-the-shelf test management software, programming the allocation of shared instrumentation among multiple test sockets running a parallel test system required that you add a large amount of low-level synchronization code to test programs. Critical sections and mutexes often were intertwined with the actual code, making it difficult to program or reuse sections in future test systems.

By implementing parallel test systems that leverage many of the built-in features in NI TestStand, you can effortlessly control the sharing of instruments and synchronize multiple devices under test. You can use synchronization step types and configurable test properties at the individual test level to manage resource sharing between tests in a sequence. The synchronization step types used in test sequences often include lock, rendezvous, queue, notification, wait, and batch synchronization step types. Figure 8-5 shows how you can use a lock step while testing two UUTs.

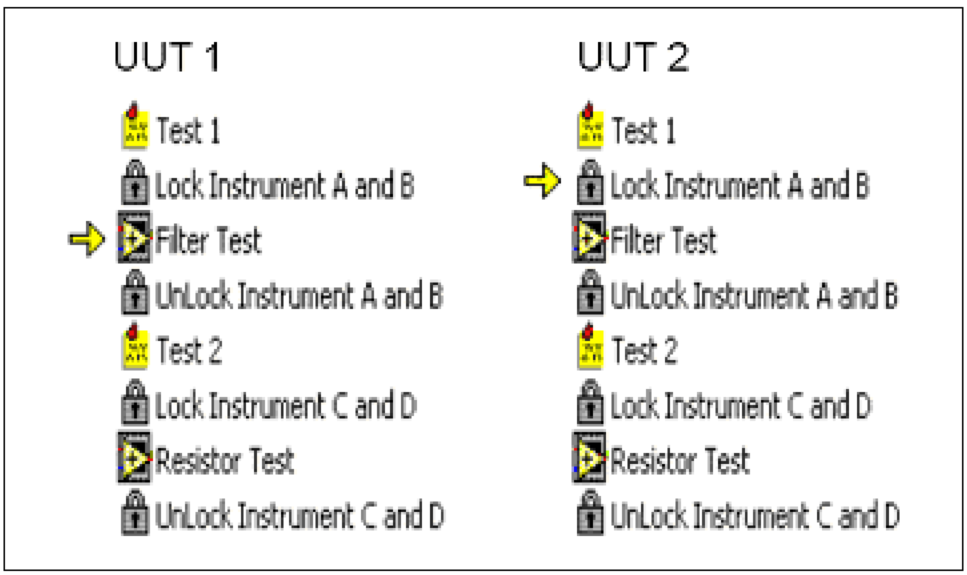

Figure 8-5. This example test sequence uses a combination of lock step types to prevent multiple tests from trying to access the same instrument simultaneously.

Conclusion: Maximizing Throughput in an Automated Test System

When developing a system were throughput is critical, you should choose the highest-throughput bus, select software that takes full advantage of the latest processors, use hardware synchronization, and design a system architecture that supports parallel test and resource sharing. The overriding theme of these four strategies is to choose a hardware and software platform that takes full advantage of the latest PC technology, such as PXI and LabVIEW.

9

Maximizing Accuracy in Automated Test Systems

When designing automated test systems, maximizing accuracy is usually a key concern. Determining how to maximize accuracy can be difficult. Most test engineers turn to the data sheets for the instruments they are evaluating with the hope that these documents provide all of the answers. However, other factors are just as important in maximizing accuracy in your automated test systems.

This paper provides you with five steps that you can follow to maximize the accuracy of your automated test systems. The five steps are the following:

1. Understand instrument specifications
2. Consider calibration requirements
3. Be aware of the operating environment
4. Use proper fixturing
5. Take advantage of synchronization

Understand Instrument Specifications

When evaluating the accuracy of an instrument, the data sheet is a valuable resource. However, it is important to understand that different instrument vendors oftentimes specify measurement accuracy using either different terminology or similar terminology with different meanings. Thus, it is important to have a clear understanding of all the parameters involved in defining the characteristics of an instrument. Often the terms resolution, precision, and accuracy are used interchangeably, but they actually indicate very different entities. Although common sense indicates that a 6½-digit digital multimeter (DMM) must be accurate to the 6½-digit level, this may not be the case. The number of digits can simply relate to the number of figures that the meter can display and not to the minimum distinguishable change in the input. You need to verify that the instrument sensitivity and effective resolution are enough to guarantee that the instrument will give you the measurement resolution you need.

For example, a 6½-digit DMM can represent a given range with 1,999,999 counts or units. But if the instrument has a noise value of 20 counts peak-to-peak, then the minimum distinguishable change must be at least 0.52 x 20 counts because resolution, which is the smallest amount of input signal change that an instrument can detect reliably, is equal to counts or volts of Gaussian noise multiplied by 0.52. Thus, the effective number of digits (ENOD) for this particular 6½-digit DMM is:

$$\text{ENOD} = \log_{10}(2 \bullet (1,999,999)0.52 \times 20) = 5.585 \text{ digits}$$

As you can see, the number of digits listed in the data sheet for a DMM is an important piece of information, but it should not be considered the ultimate or only parameter to take into account. By knowing the measurement accuracy and resolution requirements for your automated test systems, you can compute the total error budget of the instruments you are considering and verify that they satisfy your needs. Moreover, do not hesitate to ask vendors to clarify the meaning of the specifications in data sheets because not knowing the true performance of your instruments could lead to costly errors.

Consider Calibration Requirements

Regardless of the accuracy of the instruments you select for your automated test systems, it is important to realize that the accuracies of the electronic components used in all instruments drift over time. The effects of time in service as well as environmental conditions add to this drift. As time progresses, changes in component values cause greater uncertainty in your measurements. To resolve this issue, your instruments must be calibrated at regular intervals.

External calibration is the comparison of instrument performance to a standard of known accuracy. The result of an external calibration may be documentation showing the deviation of a measurement from the known standard, but most often it also includes adjusting the measurement capability of the instrument to ensure that its measurement accuracy is within vendor-provided limits. Many vendors provide graduated accuracy tables (see Figure 9-1), which offer a clear uncertainty profile from the last external calibration of an instrument.

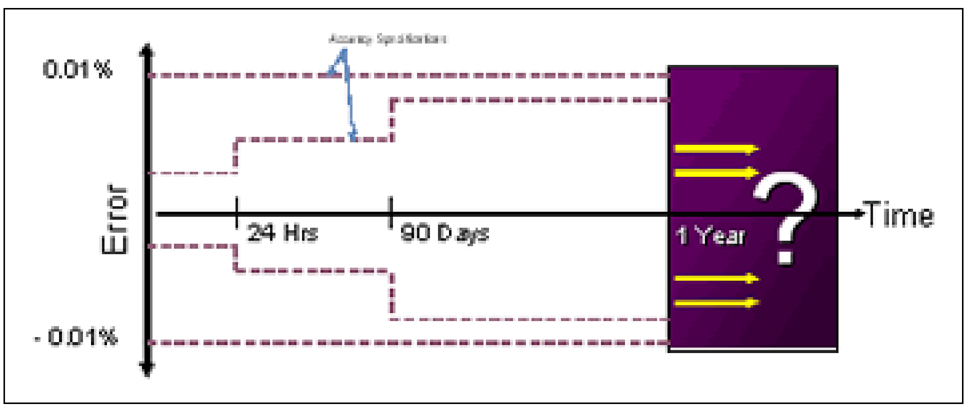

Figure 9-1. Graduated accuracy tables provide you with a clear uncertainty profile from the last external calibration of an instrument.

To have an instrument externally calibrated, you can send it back to the vendor, or you can send it to a calibration or metrology laboratory. Additionally, you may have external calibration capabilities at your facility. Regardless of how you will externally calibrate your instruments, it is important to realize that external calibration intervals for a particular type of instrument are not always the same for different vendors. One vendor may have an external calibration interval of one year for a function generator, while another vendor's external calibration interval for a function generator with equivalent or better accuracy specifications may be two years. Choosing the second instrument reduces the cost of maintaining your

automated test system. When selecting instruments, be careful to consider the external calibration intervals (see Figure 9-2).

PFI 1	SMB (Jack)
Digital data and control	68-pin VHDCI Female Receptacle
Environment	
Operating temperature	0 to +55 °C (Meets IEC-60068-2-1 and IEC-60068-2-2)
Storage temperature	-25 to +85 °C (Meets IEC-60068-2-1 and IEC-60068-2-2)
Relative humidity	10 to 90%, noncondensing (Meets IEC 60068-2-56)
Calibration	
Self-calibration	DC gain and offset
External calibration interval	2 years
Certifications and Compliances CE	
CE Mark Compliance	

Figure 9-2. Not all instruments of the same type have equivalent external calibration intervals. When selecting instruments, consider the external calibration intervals.

In addition to external calibration, instruments for some vendors include self-calibration functionality. Instruments that offer self-calibration include hardware resources such as precision voltage references so you can quickly calibrate the instrument without removing it from the test system or connecting it to external calibration hardware. Self-calibration is not a replacement for external calibration, but it does provide a method of improving instrument measurement accuracy between external calibration intervals.

Maintaining properly calibrated instruments reduced measurement errors, improves consistency between measurements, and provides you assurance that you are making accurate measurements.

Be Aware of the Operating Environment

Not all instruments have the same environmental specifications. The storage and operating temperature and relative humidity specifications can vary between vendors. Your automated test systems may be in an office-type environment where temperature and humidity are tightly controller, but they may be in a factory or other industrial setting. At the very least, it is important to be aware of the environmental specifications for your instruments and understand how they can affect measurement accuracy.

As an example, traditionally, DMMs are external calibrated at a particular temperature, and this calibration is characterized and specified over a limited temperature range, usually ±5 ºC (or even ±1 ºC in some cases). Thus, whenever the DMM is used outside of this temperature range, its accuracy specifications must be derated by a temperature coefficient, usually on the order of 10% of the

accuracy specification per ºC. At 10 ºC outside of the specified range, you may have twice the specified measurement error, which can be a serious concern when absolute accuracy is important.

Keeping the environmental temperature of a precision instrument within ±5 ºC can be challenging in a production environment, or in an automated test system composed of multiple instruments. Instruments in a system are subject to temperature rise caused by inherent compromises in air circulation and other factors. If the changes in temperature exceed these limits, and tight specifications are required, then recalibration is also required at the new temperature. Take, for example, the 10 VDC range on traditional DMMs. A DMM may have an accuracy of:

1-year accuracy: (35 ppm of reading + 5 ppm of range) for T = 23±5 ºC

In this specification, if you apply 5 V to the input, the error is:

(35 ppm of 5 V + 5 ppm of 10 V) = 225 µV, for the
temperature range 18 to 28 ºC

This is the traditional method of specifying accuracy. If the ambient temperature is outside of the 18 to 28 ºC range, the user needs to derate the accuracy using the temperature coefficient (tempco). With the traditional method, the only way to achieve the specified accuracy outside of the 18 to 28 ºC range is to fully recalibrate the system at the desired temperature. Of course, this process is often impractical and expensive. In the same example, if the DMM ambient temperature is 50 ºC, perhaps due to stacking of many instruments in a rack with limited airflow, and the tempco is specified as:

tempco = (5 ppm of reading +1 ppm of range)/ºC

Then the additional error is:

22 ºC x tempco = (120 ppm of reading + 22 ppm of range) = 1045 µV

This error at 50 ºC ambient temperature is nearly 5X greater than the specified 1-year accuracy. To eliminate errors caused by the operating environments of your automated test systems, instruments from certain vendors include features such as self-calibration (as discussed previously). This feature results in highly accurate, ultrastable instruments at any operating temperature, even well outside of the traditional 18 to 28 ºC range. To revisit the previous DMM example, the additional error introduced by temperature coefficient using a National Instruments PXI-4070 DMM with self-calibration would be fully covered in its 90-day and 2-year specifications and would be:

tempco with self-cal: < (0.3 ppm of reading + 0.3 ppm of range)/ºC
(already accounted for in the specification)

Condition	Traditional 6½ (1-Year)	NI PXI-4070 (2-Year)
Measurement within 18 to 28 ºC	225 µV	130 µV
Measurement at 50 ºC without self-calibration	1045 µV	470 µV
Measurement at 50 ºC with self-calibration	1045 µV (no self-calibration available)	130 µV

Figure 9-3. To eliminate errors caused by the operating environment of your automated test systems, instruments are available with self-calibration.

Use Proper Fixturing

Connecting your automated test system to the device under test (DUT) may be as simple as cabling from the instruments to a breakout box or screw terminals and connecting to the DUT for systems with less than 50 test points or only a few instruments. For larger systems with hundreds of test points, multiple instruments, reconfigurable system requirements, and/or frequent connects/disconnects, an approach such as a mass interconnect system is usually required.

In either situation, it is important that your fixturing be composed of cabling designed to maximize measurement accuracy. Low-quality cabling can have a significant negative impact on the accuracy of your automated test systems. Cabling that is designed to maximize measurement accuracy exhibits characteristics such as low leakage and low thermal emf (see Figure 9-4).

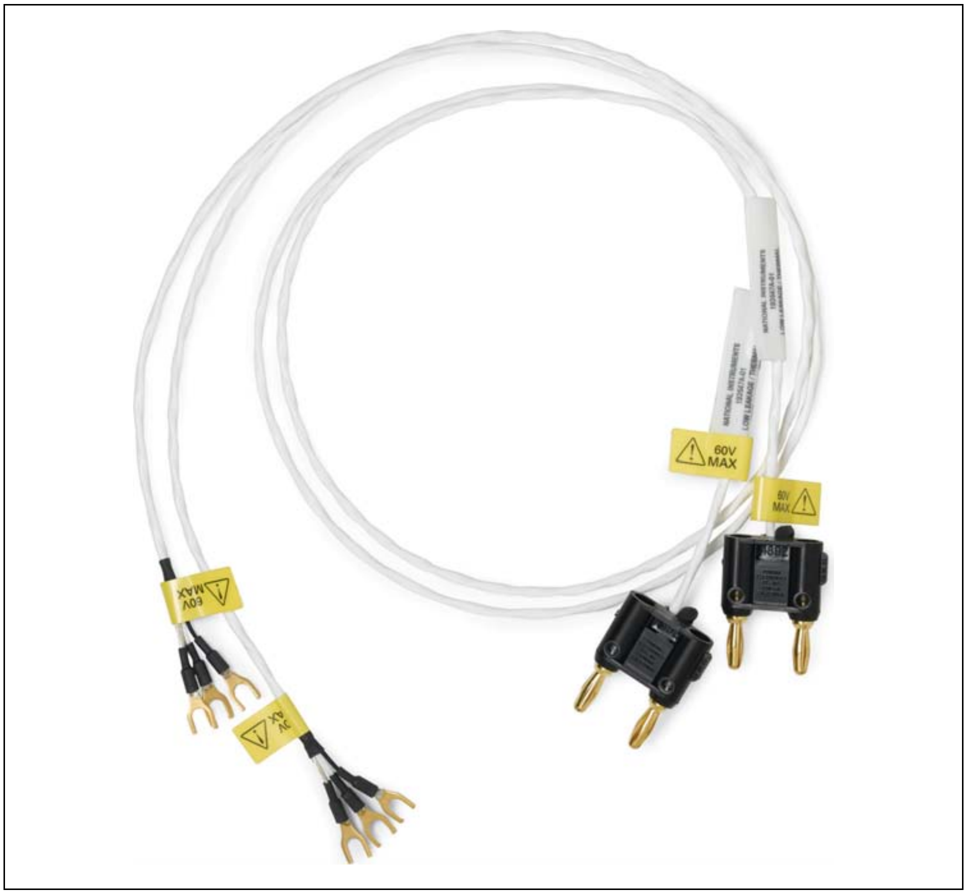

Figure 9-4. Low-leakage, low-thermal-emf cables help to maximize accuracy in your automated test systems.

As discussed previously, a mass interconnect system is a mechanical fixture designed to facilitate the connection of a large number of signals either coming from or going to a DUT. This usually entails some mechanical enclosure through which all signals are routed from instruments (typically in a rack) to the DUT, making it easy to quickly change out DUTs (see Figure 9-5). A mass interconnect system also protects the cable connections on the front of the instruments from repeated connect/disconnect cycles. Instrument cable connections that have experienced excessive connect/disconnect cycles are subject to wear and damage, which can degrade measurement accuracy.

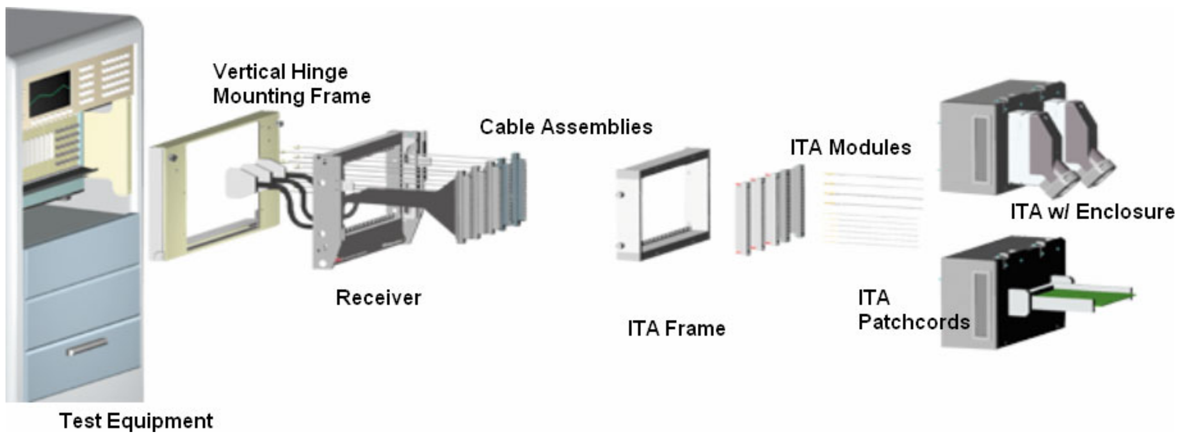

Figure 9-5. A mass interconnect system is a mechanical fixture designed to facilitate the connection of a large number of signals either coming from or going to a DUT (image courtesy of Virginia Panel Corporation).

Take Advantage of Synchronization

Another aspect of accuracy in automated test systems is phase accuracy – the degree to which the timing of signals being acquired and generated are precisely correlated. Synchronization, specifically hardware synchronization, minimizes the skew between instruments, which maximizes correlation. For example, if your automated test system includes two oscilloscopes that simultaneously acquire data from the DUT, unless the oscilloscopes use synchronized start triggers and sample clocks, it is almost impossible to compare the phases of the acquired signals.

Another example of using hardware synchronization to maximize phase accuracy in your automated test system is phase-locked looping the sample clocks for an arbitrary waveform generator (ARB) and oscilloscope to the same stable reference clock in a stimulus-response test. If precise hardware synchronization is not employed in a stimulus-response test, a fractional number of cycles of the analog wave being generated by the ARB will be acquired by the oscilloscope. When the acquired sine wave is analyzed using an FFT, spectral leakage is present as "skirts" in the spectrum, as is illustrated by the white trace on the graph in Figure 9-6. The use of phase-locked looping synchronization eliminates a fractional number of cycles from being acquired by the oscilloscope. This, in turn, eliminates the spectral leakage, as is illustrated by the red trace in Figure 9-6.

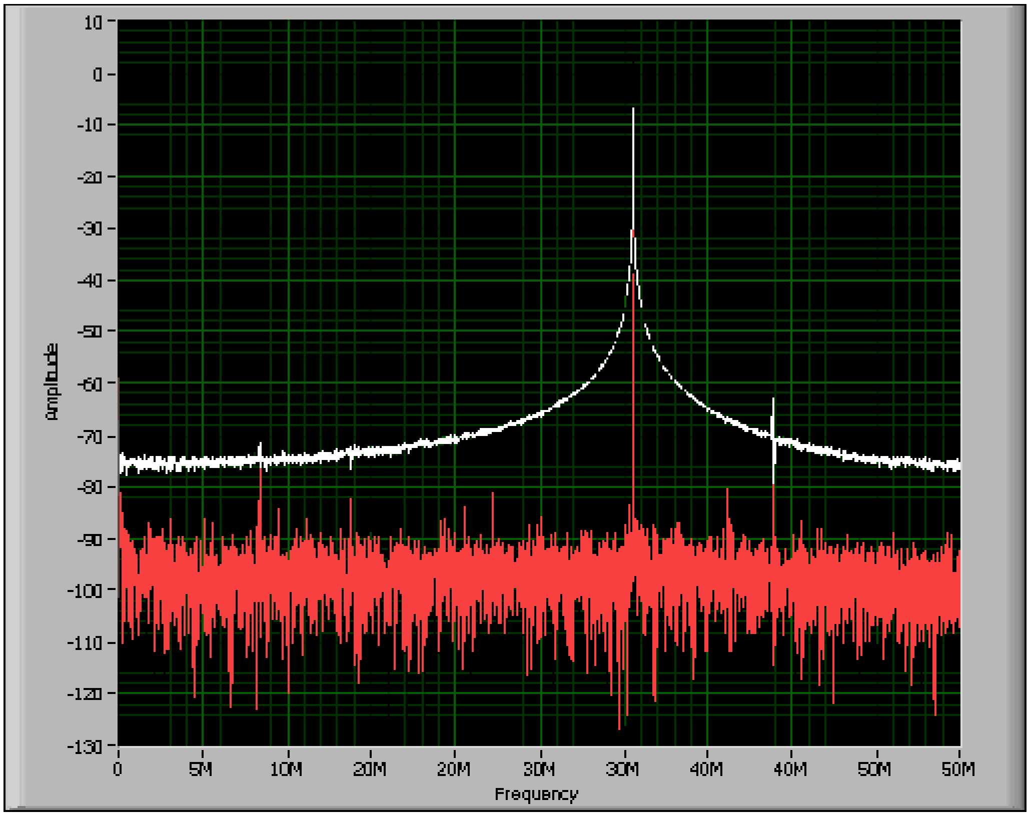

Figure 9-6. Synchronization improves phase accuracy in your automated test systems. For example, this graph illustrates hardware synchronization eliminating spectral leakage in a stimulus-response test.

Test platforms provide significantly different levels of hardware synchronization. Some offer limited functionality, and others, such as PXI, offer very sophisticated hardware synchronization resources. PXI has a high-performance timing and synchronization bus built into the backplane, which eliminates the need to use external cabling between instruments. Using the integrated PXI timing and synchronization bus, the instruments in your automated test systems can be synchronized to the sub-nanosecond level.

Conclusion: Maximizing Accuracy in Automated Test Systems

Most likely, maximizing accuracy is important to you when designing automated test systems. The five steps discussed in this white paper – understand instrument specifications, consider calibration requirements, be aware of the operating environment, use proper fixturing, and take advantage of synchronization – provide you with a roadmap for maximizing accuracy in your automated test systems.

10

Designing and Maintaining a Test System for Longevity

Becoming a technology laggard in the 21st century is far from inconceivable. As human dependence on gadgets grows, the demand for developing new technology intensifies. Cell phones are prime examples of products that are continually evolving. The typical cell phone begins production, goes into distribution, and becomes obsolete all in less than three years. Such rapid technological advances mandate that test systems built to examine products with fast life cycles be expandable to sustain product growth over the long haul. These systems must not only be modular and flexible to support copious tests that may vary between product models, but they also must be scalable to accommodate a larger number of test points. Most importantly they must be cost-effective and easy to use so as to reduce development time.

For a single test system to have all of the above listed features would involve the use of scalable software coupled with modular and expandable hardware. The goal of this document is to list software, hardware, and maintenance/support considerations that can prepare engineers to design quality test system architectures with extensive life spans.

Software Considerations

The software component of a test system is used to deploy the hardware and display processed data to the end user. The ideal software package should minimize the effort required when developing and expanding the test program and, thus, streamline the productivity of software engineers. This section lists the factors to consider when choosing software tools for designing long-term test system architectures.

- Modular Software Framework Optimized for Scalability
- Code Reuse Features
- Supports Platform Independence

Test systems built for longevity usually require modifications during their life spans. To make the process of incorporating these changes into the test program less time-consuming, the software must be scalable. Examples include easy-to-use application programming interfaces (APIs) that minimize the need to learn hardware caveats and the abundance of example code that serves as a starting point for any application. It must also minimize the effort required to perform new analysis on data acquired from the hardware by providing a host of precoded analysis functions that perform complex mathematical processing.

Test systems often need to be expanded either to accommodate more test points that may be required for testing newer models of a device under test (DUT) or to perform new tests on an existing DUT. Thus, software used to write test applications must not only be able to support hardware expansion but it must also simplify the process of modifying existing test code by maximizing code reuse. Some software tools do this by providing interactive configuration utilities that convert user input into data that is then used by the test code. By taking advantage of such utilities, users can expand the test program by making changes to the configuration file rather than to the actual code, thus reusing existing code. Another way that software can maximize code reuse when system hardware changes occur is to support drivers that use a standard communication protocol to program multiple instruments with the same API. Interchangeable Virtual Instruments, or IVI, are a prime example of such drivers, as shown in figure 10-1. IVI drivers are sophisticated instrument drivers that feature increased performance and flexibility for intricate test applications that require interchangeability, state caching, or simulation of instruments. These drivers minimize the need for hardware engineers to learn new protocols or hardware commands that may vary among instruments from different vendors.

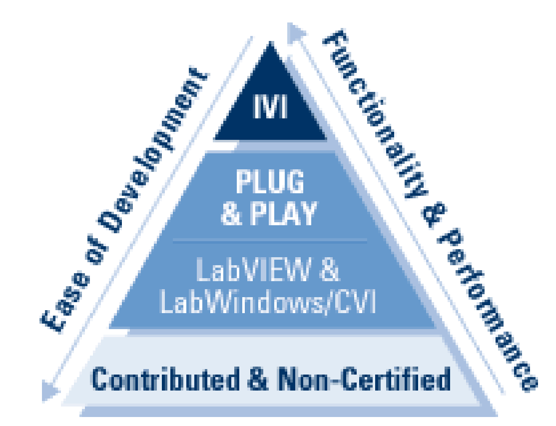

Figure 10-1. An IVI driver architecture maximizes code reuse and works with multiple hardware platforms.

To achieve interchangeability, the IVI Foundation has defined specifications for the following eight instrument classes: digital multimeter (DMM), oscilloscope, arbitrary waveform/function generator, DC power supply, switch, power meter, spectrum analyzer, and RF signal generator. For the purpose of illustrating the advantages of IVI, consider a DMM/switch system that is being expanded. Because of higher throughput demands, an additional DMM might be needed. However, because of cost constraints, a cheaper model of the instrument is chosen from a different vendor. If the test program for the system were written using IVI drivers, the code to deploy the first DMM could be reused to deploy the second one as well. For more information on IVI, visit the IVI page.

Lastly, systems built to sustain long-term technological advances must be flexible and use application development environments (ADEs) that can withstand structural changes by working with multiple hardware and software platforms. If

the ADE does not support these multiple platforms, programmers need to use different ADEs for different projects and spend time porting existing intellectual property from one platform to the other. Consider an older application built to work on Windows 98 that needs to be transported over to a newer PC that runs Windows XP. If the ADE used to develop the code did not support Windows XP, the test program would have to be completely reconstructed. Similarly, imagine that your application needs change and you must add a new measurement device. If your development software does not support this new hardware, then you may have to substantially change the application. Both scenarios would exponentially increase development time and significantly affect the efficiency of the test engineer. Therefore, a software package well-suited for designing test system architectures for longevity must support multiple vendor hardware platforms and numerous OSs. Moreover, it must be built on an adaptable framework that simplifies the process of adding support for new hardware and OSs when needed. Software with these features helps minimize the time needed to modify existing test code when new hardware or tests are introduced in the system, thus increasing application longevity.

Hardware Considerations

Consider a DUT that is a product model of a gadget that is constantly evolving – for example, the Apple iPod. Older versions of this device such as the iPod mini (now obsolete) were audio-only devices. Newer iPods can play sound as well as video. An evaluation of past trends and the wide adoption of the iPod show that it is possible that future versions of the device will incorporate additional features. Testing complex and evolving devices such as the iPod requires a flexible hardware platform that is constantly growing. This section lists the factors to consider when choosing hardware platform for designing long-term test system architectures.

- Open Industry Standard
- Modular and Flexible for Easy Expansion
- Ability to Meet Various Test Needs

With technology evolving, test needs are bound to become more complex. Therefore, it is crucial that the hardware platform chosen to build the test system is continually growing as well. Such a platform eliminates the need to completely redesign a test system architecture when its maximum capacity is reached. Open industry platforms, such as PXI, VXI, and GPIB, are usually good examples of this type of hardware. This type of platform reap the benefits of multivendor adoption, which include constant growth and innovation. Increased channel-count, voltage, current, speed, and accuracy specifications are just a few areas in which vendors that manufacture open-standard hardware compete with each other for industry recognition. This competition fosters healthy platform growth and a constant supply of new products to meet the needs of advanced test systems.

Open standards that are modular further reduce system costs by maximizing component reuse. Their flexibility eliminates complexities when system expansion is needed. Because test systems built for longevity are often expanded to incorporate more I/O points during their life spans either to test new DUTs or to add testing features to existing DUTs, a flexible hardware foundation is crucial.

Hardware chosen in test systems that need to last several years must have the ability to perform a vast number of tests. To do so, it must have a large portfolio of products that can perform tests on analog, digital, and RF signals at various levels with accuracy and speed. With the iPod example, future versions of the device might need a test system with expanded channel count and the ability to perform video, audio, and maybe even RF (in the case newer iPods that play radio stations). Open-standard platforms, because of their vast product offering and continual growth, are more likely to fulfill these requirements.

Maintenance and Support Considerations

Combining the right complementary hardware and software tools can facilitate the design of long-term test system architectures. However, even the right tools are rendered useless if they are not supported by their manufacturers. Systems that have longer life spans need to be updated and calibrated for accuracy. Furthermore, these systems need software updates to keep up with changing software requirements such as updating the OS of the application from a previous Windows version. Lastly, hardware components in systems built for longevity have a higher likelihood of needing repair. These factors mandate that products used in long-term test systems come from manufacturers that provide support services such as:

- Technical support (phone and Web)
- Calibration services
- Software subscriptions
- Repair
- Warranties

Engineers should preferably choose hardware and software that have longer life spans than the system itself. If this is not possible, they must ensure that these products are manufactured by vendors who have a strong track record for providing drop-in replacements and support for obsolete products. Without vendor support, technical issues and unforeseeable problems in the system can be hard to troubleshoot. This can sometimes cause unwanted delays and, in more extreme circumstances, require the construction of an entirely new system.

NI Solution for Building and Maintaining Test Systems for Longevity

National Instruments is committed to providing products and services that engineers can use to build test systems for the long term. The recommended solution is a five layer system architecture that was introduced in the executive summary of this guide. This section will highlight the main products and services that ensure support over the long-term.

LabVIEW Graphical Programming Environment

From a software perspective, the NI LabVIEW platform forms the core of all NI software. As a graphical programming language, NI LabVIEW is based on three basic steps that can be used to construct any test program – acquire, analyze, and present.

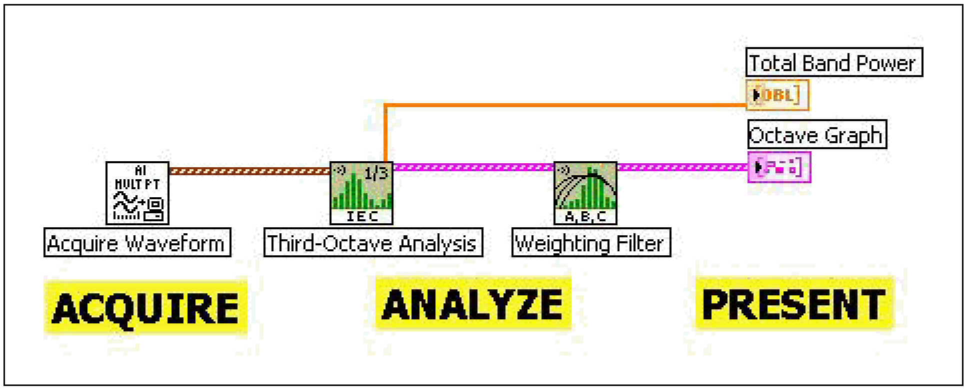

Figure 10-2. There are three steps to programming in LabVIEW.

For acquisition, LabVIEW has APIs, as shown in figure 10-2, for programming various NI hardware devices such as DMMs, RF instruments, digitizers, programmable power supplies, and multifunction data acquisition devices. Because these APIs are the result of cohesive efforts by NI hardware and software R&D engineers, they follow the same programming sequence. This consistency makes programming various instruments in LabVIEW menial and eliminates the need for programmers to learn the nuances of each instrument. Most importantly, it minimizes development time. To simplify the acquisition process even more, LabVIEW uses configuration wizards and follows the dataflow paradigm, making it simple and intuitive for first-time users to write test code.

In addition to acquisition, the LabVIEW platform offers hundreds of built-in analysis functions that cover different areas and methods for extracting information from acquired data. Programmers can use these functions as is or modify, customize, and extend them to suit a particular need. These functions are categorized in the following seven groups: measurement, signal processing, mathematics, image processing, control, simulation, and application areas.

LabVIEW also has a vast number of abstraction features such as charts, graphs, knobs, and the ability to export data to data management software such as Microsoft Excel or NI DIAdem, which help present analyzed data to the user in a clear fashion. Data presentation also can be done between locations by making use of the LabVIEW ability to publish data to the Web or log information to a database. Cohesively, these features make LabVIEW scalable and ideal for building long-term test system architectures.

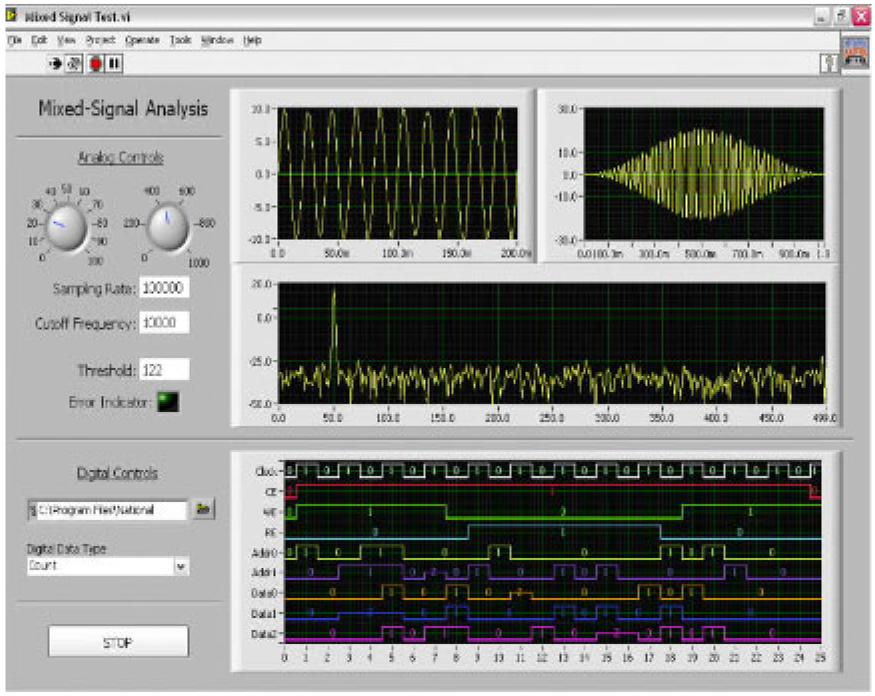

Figure 10-3. LabVIEW is an intuitive graphical programming language that provides tools such as charts, graphs, and buttons to abstract information for the user.

LabVIEW also comes with built-in features and add-on modules and utilities to help reuse existing code. These features include interactive configuration wizards such as Measurement & Automation Explorer and NI Switch Executive, which convert user input into data that is fed into LabVIEW test code, as shown in figure 10-4. Using such graphical configuration wizards, programmers can add hardware to programs by making changes to the configuration file rather than to the test program. This maximizes code reuse and increases development efficiency. In addition to configuration tools, the LabVIEW platform offers users a host of IVI class drivers to help maximize code reuse in the test program. For more information on IVI, visit the IVI page.

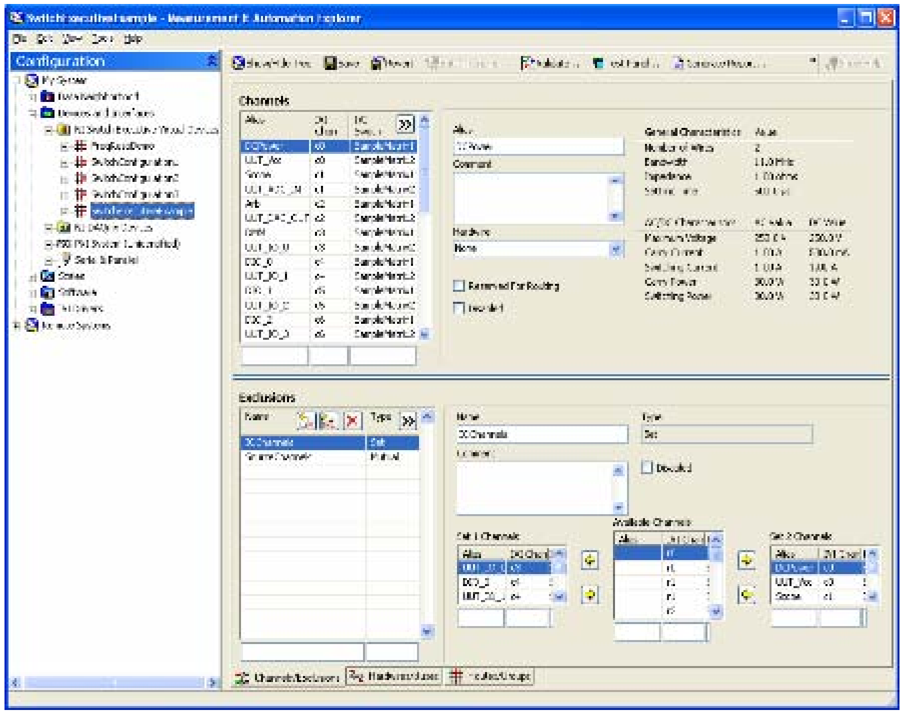

Figure 10-4. Engineers can use NI Switch Executive to configure and deploy a 9,216-crosspoint switch matrix within minutes.

The last requirement that software packages need to meet is platform independence. Complex test systems sometimes require precision instruments. When an NI solution for such instruments does not exist, programmers can make use of more than 5,000 FREE instrument drivers to program instruments from numerous vendors including Agilent, Fluke, and Keithley. Learn more on the Instrument Driver Network page. In this way, programmers can use the LabVIEW ADE to program instruments from multiple vendors, making it independent of the hardware platform. LabVIEW also supports multiple OSs and can be deployed on Windows, Mac OS, Linux®, real-time OSs such as PharLap, and even FPGA. The flexibility of LabVIEW is one of its biggest assets. Using traditional text-based languages programmed in Windows, on an FPGA, and on an embedded processor would require engineers to learn three completely separate languages (for example, C, VHDL, and TI Code Composer Studio™). Using LabVIEW, engineers can deploy applications on all three targets in the same environment, maximizing efficiency and minimizing time spent learning new tools.

For a test system to function, its software and hardware need to work in a congruent fashion by complementing each other. It is just as important to select the right hardware platform as it is to pick a powerful software tool when designing a test system for longevity. Using a flexible open hardware platform, such as PXI, minimizes the need for test system replacement as product life cycles end and technological advances are made. This increases test system longevity.

PXI, an Open Industry Standard Platform

PXI is an open-standard platform used by various test and measurement industry leaders who are all members of the PXI Systems Alliance. Currently, the more than 70 companies worldwide that are PXI Systems Alliance members share a common commitment to providing an open platform equipped for a variety of applications, from machine control to automated test. Growth of PXI modules has been rapid since the adoption of the PXI standard in 1998, and today more than 1,200 PXI products are available. In addition to a vast product offering, PXI is also expected to continually grow for the foreseeable future, as shown in figure 10-5. These PXI characteristics make it suitable for use in long-term test systems.

Test and Measurement Revenue Forecast

Instrument Type	Revenue ($ Millions) 2004	Revenue ($ Millions) 2011	Compound Annual Growth Rate
Electronic Counters	55.0	69.6	3.4%
Signal Generators	516.0	687.9	4.2%
Logic Analyzers	217.1	260.8	2.7%
Oscilloscopes	1019.9	1245.7	2.9%
Network Analyzers	282.9	380.2	4.3%
Spectrum Analyzers	589.1	813.5	4.7%
Power Meters	73.3	97.7	4.2%
Multimeters	447.8	529.4	2.4%
LCR Meters	49.5	70.1	5.1%
PC-Based	140.0	231.6	7.4%
VXI-Based	236.6	315.0	4.2%
PXI-Based	*118.1*	*564.9*	*25.1%*
AWG/Function Generators	93.7	141.3	6.0%
Total	3839.9	5407.7	5.0%

Source: Frost & Sullivan

With overall T&M revenues expected to grow by 5% per year, the greatest demand seems to lie in computer-based systems.

Figure 10-5. The PXI platform has enjoyed exponential growth over the past eight years and is forecasted to continue growing.

Its modular nature also makes PXI suitable for building test systems for longevity, as shown in figure 10-6. Expanding a PXI system to incorporate more I/O points or new testing capabilities is as easy as adding one of the more than 1,200 available modules to an empty slot in a PXI chassis, which is the outer frame of the system. Because all instruments are powered by the chassis, components such as the chassis fan and the power supply are reused to the maximum.

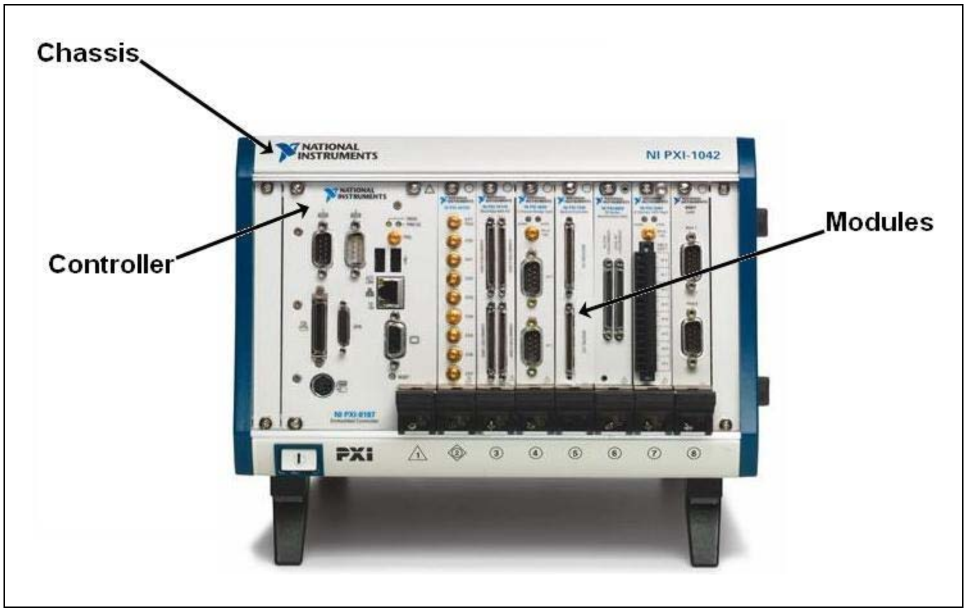

Figure 10-6. The modular PXI platform makes system expansion easy.

In addition to being an open standard and modular in nature, PXI has a wide span that addresses the needs of various applications from vision testing to high voltage and current testing (high-power applications) to even RF test. PXI modules can perform these tests with accuracies of up to 7½ digits (26 bits) and rates up to 6.6 GS/s. PXI instruments are also suited for mixed-signal tests that involve both analog and digital signals. Some modules also come with built-in signal conditioning for the measurement of sensors such as thermocouples, RTDs, load cells, and strain gages. The PXI platform also has FPGA modules and real-time capability, making it suitable for test applications that need determinism. To learn more about its capabilities and offerings, visit the PXI page.

Other Modular Hardware Platforms

Other modular platforms made by National Instruments include NI CompactDAQ, a low-cost alternative to PXI, and NI CompactRIO, an FPGA-based platform designed for high-speed control and deterministic test applications, as shown in figure 10-7. By designing modular system hardware that maximizes code reuse while catering to a vast number of applications, National Instruments provides test engineers with a selection of hardware platforms appropriate for designing long-term test systems.

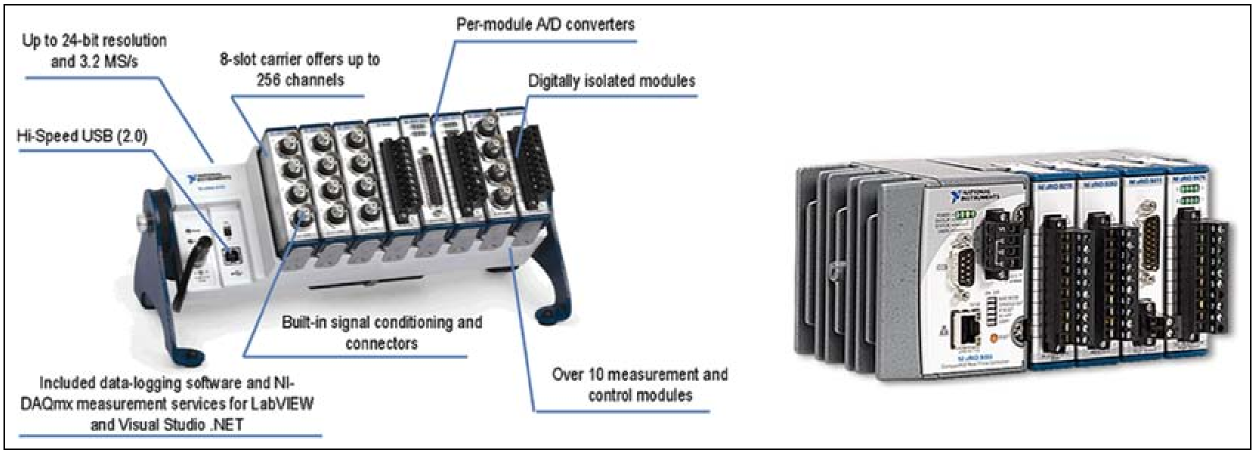

Figure 10-7. Other modular platforms manufactured by NI include NI CompactDAQ and CompactRIO.

National Instruments has a commitment to providing an array of quality hardware and software products that make designing test system architectures for longevity as elementary as possible, as shown in figure 10-8. NI makes every effort to support these products by providing calibration services; technical support via phone, e-mail, and Web; and numerous repair and warranty options.

NI also makes every effort to support obsolete products by providing technical assistance via the Web in the form of knowledgebases, tutorials and example code, and discussion forums and by offering drop-in replacements for hardware products that go "end of life." Using these services, NI hardware and software can be maintained and sustained over the long term.

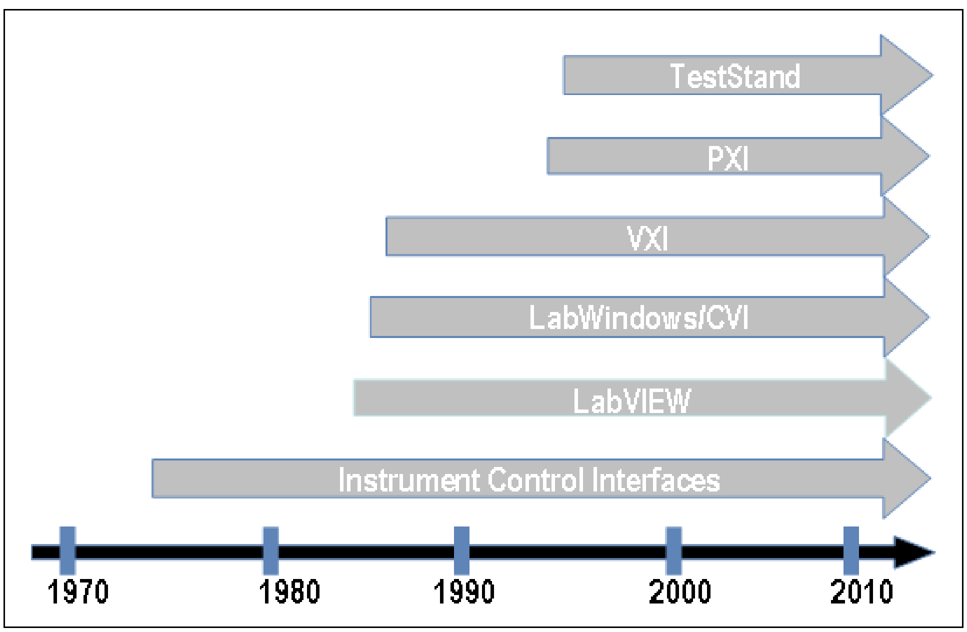

Figure 10-8. As NI releases new products, it continues to support older ones.

Conclusion: Designing and Maintaining a Test System for Longevity

You should consider software and hardware platforms that are designed to support long-term maintenance and upgrades. While hardware and software products can help build an efficient test system for longevity, it is the availability of product support from the manufacturer that determines whether it will be maintainable. In the universe of technology, what is advanced today may be archaic tomorrow – there is no avoiding that fact. With proper planning, it is possible for the test system to avoid a similar fate.

Case Studies and Customer Applications

11

Software-Defined Radio Architecture for Communications Test

Bluetooth, WiMAX, cdma2000, ZigBee, GSM, EDGE, RFID -- the list of wireless and communications standards continues to grow at an unprecedented pace, as shown in Figure 11-1. At the same time, viewing football highlights on V CAST and obtaining location data from Google Earth are becoming commonplace, fueled by the likes of Microsoft, Vodafone, and Google. Given this insatiable demand for more data bandwidth and the fact that wireless communications are now outpacing land communications in many countries, the large challenge ahead for mobile communications becomes meeting this demand effectively.

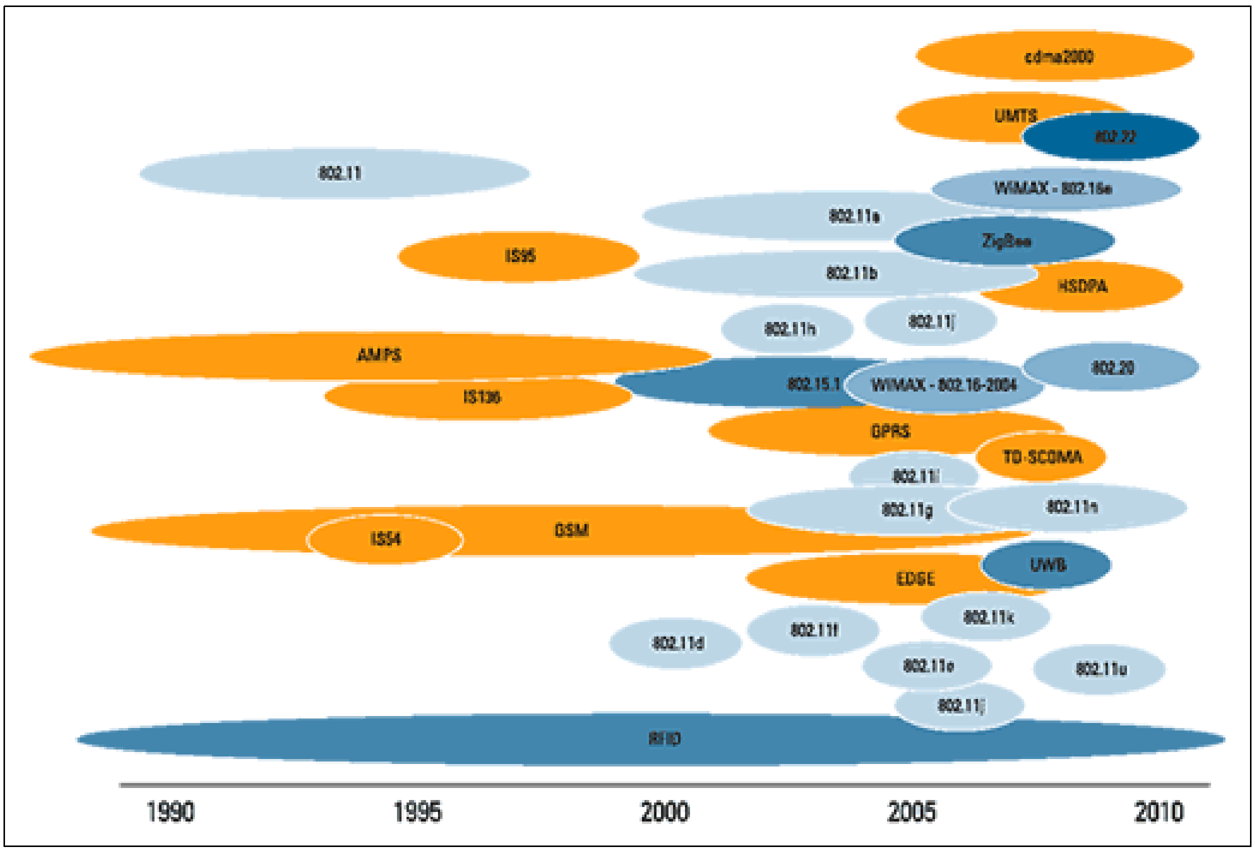

Figure 11-1. The increasing demand for data creates a wireless and communications standards "log jam."

In addition to the demand for multiple wireless standards, industry is driven by the ever-present pressure to quickly get new products to market, and research and design are outpacing test. Manufacturers release ZigBee and 802.11n devices before the standards are complete. Predefined standard test systems from

stand-alone instrument manufacturers no longer exist. This is attributable to the fact that the traditional cycle of releasing a wireless standard, prototyping devices among lead users, and developing test equipment for mass commercial use is too time-consuming.

The demand for devices that incorporate multiple standards and the pressure to release new products before the competition are two major reasons many engineers work with more than one wireless and communications standard. In fact, the data gathered by a National Instruments "Instruments Study" survey in Figure 11-2 illustrates that almost two-thirds of engineers designing and testing devices with wireless and communications functionality use more than one standard, with the following percentage breakdown:

- 37 percent use one standard
- 30 percent use two to three standards
- 33 percent use four or more standards

	802.11 (a/b/g/n)	BLUETOOTH	RFID	WIMAX	ZIGBEE
802.11 (a/b/g/n)		57%	67%	71%	75%
AM/FM/RDS	36%	35%	43%	14%	25%
ASTC (8-VSB)	4%	9%	10%	29%	0%
BLUETOOTH	31%		47%	57%	25%
CDMA	29%	28%	23%	43%	25%
DAB IBOC	1%	7%	7%	14%	0%
DVB	2%	2%	3%	14%	25%
EDGE	12%	13%	17%	43%	50%
GNSS (GALILEO)	1%	4%	3%	0%	0%
GPS	43%	48%	57%	71%	50%
GSM	19%	35%	23%	29%	25%
ITU-J.83B	0%	4%	7%	29%	0%
RFID	24%	30%		57%	50%
SIRIUS SATELLITE	1%	7%	10%	29%	0%
UMTS	6%	15%	10%	29%	0%
W-CDMA WITH HSPDA	11%	13%	17%	43%	25%
WIMAX	6%	9%	13%		25%
XM SATELLITE	1%	7%	10%	29%	25%
ZIGBEE	4%	2%	7%	14%	
Total Respondents	83	46	30	7	4

Figure 11-2. Almost two-thirds of engineers designing and testing devices with wireless and communications functionality use more than one standard.

Traditionally, you would need a separate stand-alone instrument for every communications standard to be tested. Each instrument has vendor-defined functionality for a particular standard. The communications measurement algorithms for the standards exist as firmware running on the embedded processor in each instrument, which means they are not user-accessible or customizable. Purchasing a new stand-alone instrument for each standard that you need to test is not productive or cost-effective. This is pushing engineers to seek flexible, out-of-the-box solutions.

Flexible Software-Defined Communications Test

One way to keep stride with wireless and communications advances is through software. You can take a software-defined approach to instrumentation by using coding and modulation software to generate and measure signals through modular, general-purpose RF instrumentation. This software-defined radio (SDR) approach to test is completely application-driven and user-defined. You can use it to leverage the software modeling and simulation software used in research and design for test and measurement. The Department of Defense (DoD) already supports this strategy.

"For the military, SDR is a transformational technology that allows the development of a truly interoperable family of radios that can communicate in any theater of operation with any allied force at any time," said Colonel Steven MacLaird, director of the Joint Systems Program and program manager for the JTRS Joint Program (SDR Forum, August 2003).

A Typical Communications System

By stepping through a simplified functional block diagram of a typical communications system, you can see how to combine communications software with modular, general-purpose RF instrumentation to create a test system that supports multiple standards. Figure 11-3 represents the major functional blocks in a typical communications system. You can use these blocks for source coding, channel coding, modulation, and upconversion on the transmit side and the reverse of this process on the receive side. A real-world communication link contains a physical channel across which the transmission occurs. Physical channel examples include air (wireless), fiber-optic, and copper.

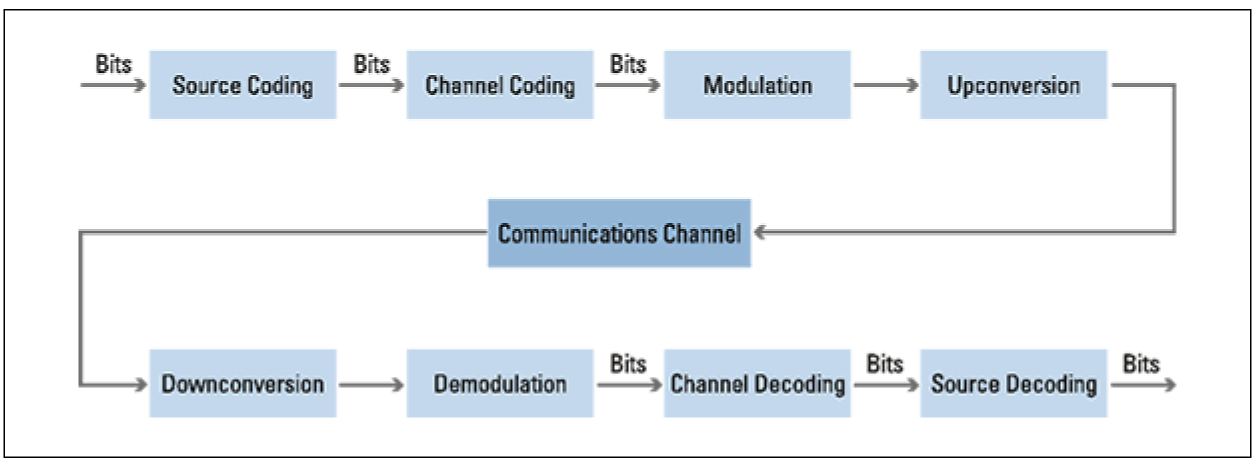

Figure 11-3. This illustration represents the major functional blocks in a typical communications system.

Source Coding and Decoding

The primary function of source coding is to represent your message in as few bits as possible to minimize resources. Source coding is similar to data compression; the smaller the message, the faster the transmission time, which translates into more efficient use of precious resources and spectrum. With source coding, you can send more information using the same bandwidth. Some of the more common source-coding algorithms include jpeg compression, zip (a combination of the LZ77 and Huffman coding algorithms), MP3 (part of MPEG-1 for sound and music compression), and MPEG-2 (used in DVDs).

Channel Coding and Decoding

Unlike source coding, channel coding can add bits to the data, which increases the message size. Added or reworked bits ensure that the original message can better withstand the effects of any channel impairments, including noise and fading, for proper decoding to obtain the original transmitted message. Many channel-coding algorithms balance the need to correctly encode and transmit data while minimizing message size.

Modulation and Demodulation

Modulation is the process of varying one or more properties (amplitude, frequency, and/or phase) of an electromagnetic wave or signal. You can use modulation to transmit information that originates at a low frequency signal to a signal operating at a higher frequency. You may wonder why you would want to transmit at a higher frequency as opposed to a lower frequency. Transmitting a baseband audio signal (from 20 Hz to 20 kHz) in a wireless fashion would require an antenna, power source, and electronic equipment of substantial proportions, which is impractical because of the large wavelength that is inversely proportional to the frequency. Therefore, if you transmit this same signal at a higher frequency, the wavelength becomes smaller, and you can reduce the size of the equipment and the amount of power you need. This fact signifies the prevalence and importance of modulation. With modulation, you can piggyback your baseband signal on a higher-frequency signal. The lower-frequency signal that contains the information or message you want to transmit is the modulating signal. The higher-frequency signal is referred to as the carrier signal because it "carries" the baseband information. The resulting combined signal is called the modulated carrier signal.

You also can use modulation when you want several signals to share the same channel or if you want to transmit more information without increasing the signal bandwidth. You achieve more efficient bandwidth use because more information can be carried in the same amount of space. You can choose a specific modulation format depending on the application and the amount of data you need to transmit. In addition to standard modulation formats, by performing modulation and demodulation in software, you can develop custom formats, which is particularly useful for proprietary and/or military applications that require custom formats.

Upconversion and Downconversion

You can use an upconverter and downconverter to shift an input frequency either up or down, respectively. The primary component of upconversion and downconversion is a device called a mixer. Mixers "multiply" two signals with different frequencies to produce a sum and difference signal.

Figure 11-4 illustrates the earlier functional block diagram of a typical communications system with National Instruments LabVIEW graphical code. The functions are for source coding, channel coding, modulation, and upconversion on the transmit side and downconversion, demodulation, channel decoding, and source decoding on the receiver side. The software is particularly suited for a PXI system, which provides the modular, general-purpose RF instrumentation required to both generate/upconvert and downconvert/acquire the communications signals.

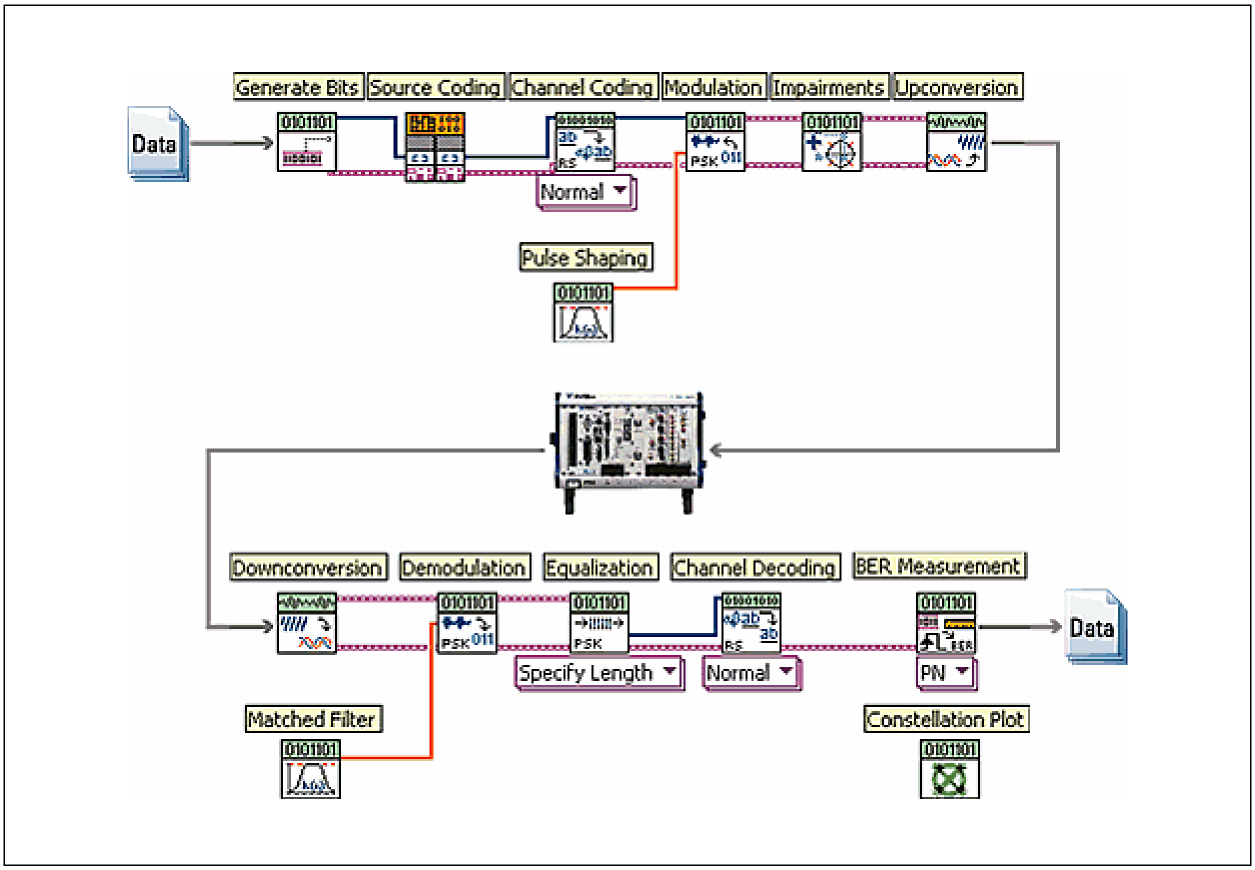

Figure 11-4. Communications software, such as NI LabVIEW, running on a PXI system, provides a flexible platform for communications test.

PXI – An Ideal Platform for Software-Defined Communications Test

There are many reasons why the PXI platform is ideal for software-defined communications test. Most importantly, it is PC-based. The functionality of PXI instruments is defined in software so a single PXI RF instrument can test multiple communications standards by simply changing the software running on the system controller. PXI controllers employing the latest dual-core processors can easily process the most complex communications algorithms.

Figure 11-5. You can use a single PXI RF instrument to test multiple communications standards by simply running different software on the system controller.

As communications standards continue to scale the amount of data transferred, it is important to base a communications test platform on a high-throughput bus to transfer the data. PXI is based on the PCI and PCI Express buses, providing up to 6 GB/s of system bandwidth and up to 2 GB/s of bandwidth to a single instrument. With this throughput, you can use PXI to perform long-duration recording of communications signals for offline analysis and the playback of previously recorded or simulated signals.

Also, with the modular nature of PXI, you can upgrade a single component of a system. For example, you can increase the performance of all of the instruments in a PXI system by upgrading to a controller with a higher-performance processor. This type of upgrade is not possible with stand-alone instruments where the embedded processor is not user-accessible or upgradable. Moreover, because PXI is a multivendor platform, the modular components of a system can come from multiple vendors. You are not locked into a single vendor, and, because all PXI products must adhere to the PXI hardware and software specifications, interoperability among different vendors is guaranteed.

Most systems that test communications must also test other device functionality and include other instruments, such as digital multimeters (DMMs), programmable power supplies, and switching. The PXI platform is general-purpose and offers

instruments for most applications and measurements. More than 1,000 PXI modules are available from the more than 68 members of the PXI Systems Alliance (PXISA).

Software-Defined Communications Systems Provide a Future-Proof Platform

The trend toward software-defined communications test systems will continue to grow. Organizations have embraced the movement because it helps them develop test systems in conjunction with standards development. Software-defined test offers the solution for current communications systems, but, more importantly, it provides a paradigm and platform for emerging and future communications systems.

12

Microsoft Uses NI LabVIEW and PXI Modular Instruments to Develop Production Test System for Xbox 360 Controllers

Author(s): D.J. Mathias, Microsoft

Product Used: LabVIEW, Modular Instruments, Oscilloscopes/Digitizers, PXI/CompactPCI

The Challenge: Developing a comprehensive, low-cost production test system for the Microsoft Xbox 360 wired and wireless controllers.

The Solution: Using a flexible, automated test system based on Microsoft Windows XP, Microsoft SQL Server, National Instruments LabVIEW, and NI PXI modular instruments to test the functional performance of the Xbox 360 controller, both wired and wireless versions.

Figure 12-1. Microsoft uses PXI and LabVIEW to ensure a quality gaming experience with the Xbox 360.

Designing Powerful Controllers for a New Generation of Gaming

In 2001, Microsoft deployed a PXI-based end-of-line functional test system for the original Xbox controller using NI LabVIEW and PXI modular instruments. The system tested device communication and monitored data packets at the bit level to verify that all controller-functional messages were within specification. The system

also monitored signals at the chip level to analyze the electrical signals for parameters such as rise/fall times, minimum/maximum voltage levels, and current draw.

In May 2005, Microsoft announced its latest innovation for digital entertainment and gaming, the Xbox 360, along with a new line of Xbox 360 wired and wireless controllers. The Xbox 360 wired controllers use a versatile, low-cost USB interface to communicate to the main game console. With the USB interface, the system easily accepts additional peripherals such as dance pads and steering wheels. The Xbox 360 controller-functional test system needed to perform similar tests to those of the original Xbox controller test system, but demanded higher-performance signal capture to qualify the signal integrity of the new controller and ensure a high-quality user experience. With the latest NI modular instruments, including the NI PXI-5124 12-bit, 200 MS/s digitizer, we met the increased functional test requirements for the Xbox 360 controller. Using the LabVIEW graphical development environment, we created more than 100 tests, implemented Ethernet communication, and incorporated a data storage interface to our Microsoft SQL Server database.

PXI Modular Instruments for Design Validation and Production Test

Using PXI instrumentation and LabVIEW, we built the test system in our Xbox 360 controller design validation lab and recently deployed it to our production line. During the validation and production cycle, the following NI PXI-based modular instruments provided us with a broad range of measurement functionality:

- PXI-5124 high-resolution digitizer for USB communication interface analysis
- PXI-4472 dynamic signal acquisition module for vibration feedback motor analysis
- PXI data acquisition modules for general-purpose analog I/O measurements
- PXI-6509 digital I/O module for general-purpose I/O control

We rapidly adapted the test system capabilities to meet our requirements for both the validation lab and production test by taking advantage of the broad range of PXI functionality, PXI modularity, and the PXI software-centric measurement approach.

The PXI-5124 high-resolution digitizer is a key component in the Xbox 360 controller end-of-line functional test system. The 200 MS/s real-time sampling rate and 12 bits of resolution on the PXI-5124 digitizer helped us verify the signal integrity of the USB communication between the controller and the Xbox 360 console with confidence. The high-resolution input and high-speed sampling rate are important features that make the digitizer a low-cost, quality solution – and a better option compared with higher-cost and lower-resolution oscilloscopes – to capture, monitor, and analyze the Xbox 360 controller USB signals,audio signals, and serial data signaling.

NI LabVIEW Interfacing with Microsoft SQL Server, TCP/IP, and ActiveX Controls

Functional test is a key component to any production line. The challenge in developing a production line functional tester is to package as many parallel test scenarios as possible within the given production cycle time. With the new functional test system for the Xbox 360 controller, we implemented a test strategy that resulted in a 100 percent increase in our test throughput per test station.

We used LabVIEW to run multiple tests in parallel to maximize test coverage during the given production cycle time, and we used the LabVIEW Database Connectivity Toolkit to connect to our Microsoft SQL Server database to store every unit under test (UUT) parameter. As each Xbox 360 controller rolls off the production line, each completed test sends more than 110 data parameters to the dedicated Microsoft SQL Server for post-test analysis to implement future production line and device enhancements. Using the integrated TCP/IP and support for embedded ActiveX controls in LabVIEW, we communicated to the USB and wireless controllers through our custom interfaces. Overall, LabVIEW helped us develop an optimized end-of-line production test system for the Xbox 360 controller with data storage to our Microsoft SQL Server, communication through TCP/IP, and programmatic interaction with ActiveX controls.

Microsoft Sees Results Using NI LabVIEW and PXI Modular Instruments

At Microsoft Corporation, we developed a versatile validation and end-of-line production test system for the Xbox and Xbox 360 controllers using Microsoft Windows XP, LabVIEW, and PXI. With the PXI-based system, we can achieve reliable production line testing and store all parameters to our Microsoft SQL Server. Using the high resolution input and high sampling rate of the PXI-5124 digitizer, we acquire our test signals with 12 bits of resolution at data rates up to 200 MS/s, which provides a low-cost automated test system. Finally, using the power of the PC, we continue to easily upgrade and maintain our system today and for future development.

For more information, contact:

D.J. Mathias
Microsoft
One Microsoft Way
Redmond, WA 98052
Tel: 1-800-MICROSOFT

13

U.S. Air Force Increases Mission-Capable Rates with PXI

Author: John Abdale, Mantech Test Systems

Product Used: Modular Instruments, PXI/CompactPCI

The Challenge: Developing, producing, and supporting test equipment for LANTIRN systems on U.S. Air Force premier fighter aircrafts.

The Solution: Using NI PC-based software and hardware to lower costs and reduce size of test systems by 50 percent.

Figure 13-1. LANTIRN system on an F-16 on the flight line. Photograph courtesy of Lockheed Martin.

U.S.Air Force Increases Mission-Capable Rates with PXI

In 2002, the U.S. Air Force awarded ManTech Test Systems a multimillion dollar contract for development, production, and support of test equipment for LANTIRN systems. LANTIRN, or Low Altitude Navigations and Targeting Infrared for Night, is a system used on U.S. Air Force premier fighter aircraft, including the F-15E Eagle and F-16 C/D Fighting Falcon. LANTIRN significantly increases the combat effectiveness of these aircraft, allowing them to fly at low altitudes, at night, and under the weather to attack ground targets with a variety of precision-guided and unguided weapons.

Using PC-Based Software and Hardware to Lower Costs

The contract charged ManTech with updating LANTIRN test systems that will be fielded at 19 Air Force locations around the world. The original LANTIRN test system dates back to the late 1980s, and was based on MicroVAX computers tied to stand-alone instrumentation. Not only was the system large, requiring seven complete racks of space, but the Air Force faced a host of reliability and maintenance problems from the growing obsolescence of test system components. In many cases, the Air Force had to reverse engineer and redesign obsolete components on the original test station. As a result of this problem, the Air Force specified and budgeted for a new test system in 2002. A major requirement of the new system was to take advantage of new commercial off-the-shelf technology, such as industry-standard, PC-based hardware and software, to reduce the size and cost of the new test system. By specifying off-the-shelf components, the military is able to search across all industries for powerful and low-cost components that are relatively easy to replace and upgrade.

Reducing Test System Size by 50 Percent

ManTech selected PXI for a portion of the test system, largely because PXI provides the advantages of commercial off-the-shelf technology, while still meeting the needs of military specifications in test programs. For example, the specification required extended operating and nonoperating environmental conditions, which National Instruments met with the new PXI-8186 Intel Pentium 4-based PXI embedded controller. This provided ManTech with the performance and cost benefits of the latest off-the-shelf processors from Intel, while also meeting the military specifications for environmental conditions of the test system. Additionally, the replacement of the obsolete MicroVax hardware with PXI and other systems dramatically increase the mission-ready time of the test system.

ManTech was also able to reduce the size of the test system from seven racks to three, a more than 50 percent reduction due in large part to the incorporation of PXI instrumentation, which includes up to 17 PXI instruments in just 4U of rack space. Commenting on the Air Force's award of the contract to ManTech, Peter D. Faulkner, vice president of government programs for ManTech Test Systems, said, "Our solution takes advantage of the many technological advances that have occurred in the automatic test industry since the LANTIRN support equipment was originally placed in service. The Air Force can expect a tremendous improvement in mission capable rates in a system that is less than half the size.

14

Sanmina-SCI Exceeds Throughput Goals with PXI Tester and Multithreaded Software

Author(s): Mike Oehrlein, Sanmina-SCI Corporation

Product: Digital Multimeters, LabWindows/CVI, Modular Instruments, NI TestStand, PXI/CompactPCI

The Challenge: Developing a compact and high-speed functional test station that performs parallel calibration on eight medical devices.

The Solution: Using National Instruments LabWindows/CVI, NI TestStand, and high-performance PXI modular instruments to develop an automated and modular production tester with database and statistical process control capability for medical device calibration.

Figure 14-1. High-Speed PXI-Based Test System for Calibrating Medical Devices

Test System Requirements

Sanmina-SCI, one of the world's leading contract manufacturers, recently developed a test application for a medical device that measures blood glucose levels by measuring the current and impedance characteristics produced from an electrochemical reaction. We needed to build a functional test and calibration system that complied with FDA regulations, calibrated the DC amplifier circuits of the measurement engine, and verified the operation of other critical support circuits contained within the device under test. Additionally, in real time, the system had to switch a variety of input loads into the measurement engine to emulate the complex task of simulating blood response. Lastly, the system had to reliably and repeatedly process up to 83,000 devices per week while maintaining a cycle time of effectively 30 seconds per device.

Because of these requirements, we chose a flexible software framework that facilitated multithreaded parallel testing; efficient communication with commercial databases; and a custom software interface with strict user management for administrators, supervisors, engineers, and manufacturing operators.

Compact, High-Speed Test Solution

The Sanmina-SCI design team knew from previous experience that a design solution based on traditional instrumentation alone could not meet test-throughput needs. For this application, the design team needed to build a hybrid test system based on PXI and GPIB modular instruments as well as a custom FPGA-controlled interface to the device under test that could test eight devices in parallel. The completed test system included eight NI PXI-4070 6½-digit FlexDMM devices, eight NI PXI-6533 high-speed digital I/O modules, an NI PXI-6508 general-purpose digital I/O module, a GPIB-based power supply, and an LCR meter. The FlexDMM delivered the throughput and accuracy the design team needed to acquire all of the voltage and current measurements on the device under test. The design team used the PXI digital devices to communicate with the DUT and write the calibration values to the onboard EEPROM.

We chose NI TestStand for test management software because it provided test sequencing of LabWindows/CVI and C# .NET modules, native parallel testing support, and a comprehensive user management framework. Its flexible and open source code operator interface also gave our design team full control over the look and feel of the operator interface. Rather than worry about thread management and synchronization, we could use NI TestStand to focus on the details of the test management process. We used LabWindows/CVI for test module development because of the built-in test and measurement functions and user interface controls, in addition to the SQL database connectivity capabilities. With LabWindows/CVI, we could create highly efficient ANSI C source code that integrated with modular instruments and GPIB. By using commercially available software such as LabWindows/CVI and NI TestStand, we kept software development time and costs to a minimum.

Exceeded Yield Expectations

The PXI-based tester provided a compact calibration system that met the high-speed throughput requirements while exceeding the initial system yield requirements by achieving yields greater than 95 percent in production.

The customer was satisfied with the completed test system and has plans to expand the system by building additional stations to satisfy product demand and adapting fixtures and software for the next-generation devices.

For more information, contact:

Mike Oehrlein
Sanmina-SCI Corporation
13000 S. Memorial Pkwy
PO Box 1000
Huntsville, AL 35807
Tel: (256) 882-4800, ext. 8587
`www.sanmina-sci.com`

References

[1] R. Harrison, A Software-Defined Platform for Current and Future Communications Systems, Instrumentation Newsletter, Q1 2006.

[2] J. Kovacs, LabVIEW and PXI Enhance Communications Design and Test, Instrumentation Newsletter, Q2 2006.

[3] Colonel S. MacLaird, Software Defined Radio Takes Step Closer To Wide-Scale Military Use, SDR Forum, August 2003.

342614A-01 Aug06

NATIONAL INSTRUMENTS™

11500 North Mopac Expressway • Austin, TX 78759-3504 USA
Tel: (512) 683-0100 • Fax: (512) 794-8411 • Email: `info@ni.com`